Simultaneous Pose and Correspondence Problem for Visual Servoing

by

Raymond Chiu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2010

© Raymond Chiu 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Pose estimation is a common problem in computer vision. The pose is the combination of the position and orientation of a particular object relative to some reference coordinate system. The pose estimation problem involves determining the pose of an object from one or multiple images of the object. This problem often arises in the area of robotics. It is necessary to determine the pose of an object before it can be manipulated by the robot. In particular, this research focuses on pose estimation for initialization of position-based visual servoing.

A closely related problem is the correspondence problem. This is the problem of finding a set of features from the image of an object that can be identified as the same feature from a model of the object. Solving for pose without known correspondence is also refered to as the simultaneous pose and correspondence problem, and it is a lot more difficult than solving for pose with known correspondence.

This thesis explores a number of methods to solve the simultaneous pose and correspondence problem, with focuses on a method called SoftPOSIT. It uses the idea that the pose is easily determined if correspondence is known. It first produces an initial guess of the pose and uses it to determine a correspondence. With the correspondence, it determines a new pose. This new pose is assumed to be a better estimate, thus a better correspondence can be determined. The process is repeated until the algorithm converges to a correspondence pose estimate. If this pose estimate is not good enough, the algorithm is restarted with a new initial guess.

An improvement is made to this algorithm. An early termination condition is added to detect conditions where the algorithm is unlikely to converge towards a good pose. This leads to an reduction in the runtime by as much as 50% and improvement in the success rate of the algorithm by approximately 5%.

The proposed solution is tested and compared with the RANSAC method and simulated annealing in a simulation environment. It is shown that the proposed solution has the potential for use in commercial environments for pose estimation.

Acknowledgements

I would like to thank all people who supported me in the creation of this thesis. I would especially like to thank my supervisors, Professor William Wilson and Professor Carol Hulls, for their guidance and advice.

Dedication

This is dedicated to the my family.

Contents

List of Figures

Chapter 1

Introduction

Pose estimation is a common problem in computer vision. The pose is the combination of the position and orientation of an object. Thus, pose estimation is the process of determining the position and the orientation of a particular object with respect to a known frame of reference.

Pose estimation is often performed by comparing features found in a model and features detected in an image of the target. How the features of the model correspond with the features in the image is usually unknown. The problem of finding this correspondence is known as the correspondence problem. Pose estimation without this correspondence is also known as the simultaneous pose and correspondence problem.

This problem often arises in the area of robotics[18]. For a robotic arm to perform any manipulation on a target object, it is necessary to know both the position and orientation of the object. This pose may be fixed or known, which is often the case in an assembly line. However, if there happens to be any error in the pose, the operation may fail. As such, it is often desirable to have a feedback system that can determine the pose of an object.

Another application lies in the area of mobile robots[21][25]. A mobile robot is not fixed in a particular position and the pose of the target object is often unknown. It will be necessary to first determine the pose before the robot can navigate to the target and perform any desired operation.

1.1 Motivation

Feedback from sensors is often required if high tolerance and flexibility is desired in an autonomous system. The use of visual images as the feedback for a control system is a technique known as visual servoing.

Visual servoing can generally be classified into two categories: image-based visual servoing(IBVS) and position-based visual servoing(PBVS). The control of

IBVS is based on the error between the current and desired feature on the image plane and does not involve any pose estimation of the target. As such, this thesis is not applicable to IBVS. Reader are referred to Chaumette and Hutchinson[4][5] for more details.

PBVS on the other hand tracks the pose of a target object with a known model over time and has been successfully demonstrated by researchers such as Wilson et al.[33]. Such a system requires the knowledge of the initial pose of the target. It then applies an extended Kalman filter on new images from the camera to update the pose of the target over time. However, without a good initial pose, the system will be unable to find a new pose of the object, creating a chicken-and-egg problem. This thesis focuses on solving the problem of obtaining a pose to be used as initialization for PBVS.

It is not necessary to achieve real time performance. However, it is impractical if the initialization requires too much time. Therefore, a time limit has to be imposed. Past researchers have mainly focused on the performance of an approach in terms of their chance of finding a good pose with little regard to the time required to retrieve such pose. This thesis analysed a few approaches and focused on their performance under a time constraint. Improvements are also proposed to reduce the time requirements on the methods.

1.2 Outline

The thesis is divided into the following chapters. The basic problem and some of the previous work are described in the Chapter 2.

Chapter 3 begins to describe the proposed solution by first tackling the subproblem of solving for pose estimation when the correspondences between the features are known. Some preliminary tests are performed to justify the reason of further research on the proposed method.

Chapter 4 provides the complete proposed solutions. It focuses on the SoftPOSIT method, but also describes the RANSAC and the simulated annealing methods. These solutions are tested against each other and compared in Chapter 5.

Conclusions and recommendations can be found in Chapter 6.

Chapter 2

Background

2.1 Problem Statement

The pose estimation problem involves finding the relative position and the orientation between the target object frame, **O**, and a global coordinate frame. In this thesis, the global frame will be taken as the camera frame, **C**. For this research, the camera frame will be located at the focal point of the camera with the negative z-axis representing the direction in which the camera is pointing.

This problem can also be viewed as solving for the transformation between two 3D frames. In the 3D space, the transformation has 6 degrees of freedom, 3 for translation and 3 for rotation. The translation can be described with the translation vector, $\mathbf{T} = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T$, and the rotation can be described by a rotation matrix, **R**. Figure 2.1 shows the effect of this transformation.

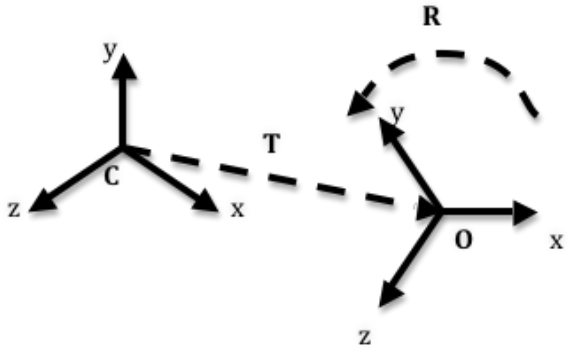


Figure 2.1: Effect of transformation

The rotation matrix, **R**, is a 3×3 orthogonal matrix with determinant equals to 1. Thus, it satisfied the constraints

$$\begin{aligned} \mathbf{R}^T\mathbf{R} &= \mathbf{I}, \\ \det(\mathbf{R}) &= 1 \end{aligned} \tag{2.1}$$

where $\mathbf{I}$ is the 3×3 identity matrix[29].

$\mathbf{R}$ can also be expressed as the concatenation of three 3×1 vectors, $\mathbf{R_i}$, $\mathbf{R_j}$, and $\mathbf{R_k}$. Giving

$$\mathbf{R} = \begin{bmatrix} \mathbf{R_i} & \mathbf{R_j} & \mathbf{R_k} \end{bmatrix}^T \tag{2.2}$$

Each of these vectors is a unit vector in the direction of the coordinate axes of the object frame when they are expressed in the camera frame.

Suppose there is a set of m feature points in the 3 dimensional space, $\mathbf{P^O} = \begin{bmatrix} \mathbf{P_1^O} & \mathbf{P_2^O} & \cdots & \mathbf{P_m^O} \end{bmatrix}$, in the object frame representing the model. Let $\mathbf{P^C} = \begin{bmatrix} \mathbf{P_1^C} & \mathbf{P_2^C} & \cdots & \mathbf{P_m^C} \end{bmatrix}$ be the corresponding set of points in the camera frame. The point $\mathbf{P_i^O} = \begin{bmatrix} P_{i_x}^O & P_{i_y}^O & P_{i_z}^O \end{bmatrix}^T$ and $\mathbf{P_i^C} = \begin{bmatrix} P_{i_x}^C & P_{i_y}^C & P_{i_z}^C \end{bmatrix}^T$ are related by the equation:

$$\mathbf{P_i^C} = \mathbf{R}\mathbf{P_i^O} + \mathbf{T} \tag{2.3}$$

Explicitly, expressing Equation 2.3 gives:

$$\begin{aligned} P_{i_x}^C &= \mathbf{R_i}^T\mathbf{P_i^O} + T_x \\ P_{i_y}^C &= \mathbf{R_j}^T\mathbf{P_i^O} + T_y \\ P_{i_z}^C &= \mathbf{R_k}^T\mathbf{P_i^O} + T_z \end{aligned} \tag{2.4}$$

The goal is to solve for $\mathbf{R}$ and $\mathbf{T}$ by analyzing a single image of the object taken by the camera.

2.2 Camera Model

The features used to describe the target object are in 3D, while the projection of a feature found in the image is in 2D. The pinhole camera model is used to define the projection of the 3D coordinates of the object features with respect to the camera frame to the 2D coordinates found in the image. This transformation from the 3D coordination to the 2D coordinate is also called a perspective projection.

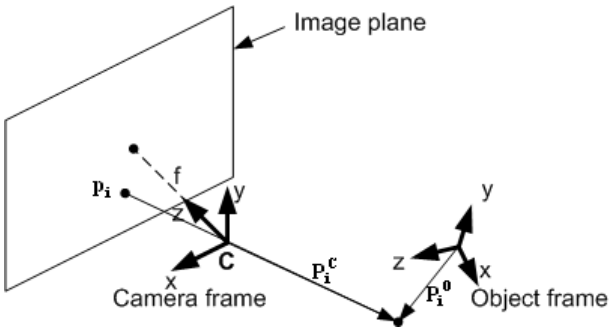


Figure 2.2: Ideal pinhole camera model

The pinhole camera model maps a 3D point onto a 2D plane with a straight line through the origin of the camera frame. A 3D point, $\mathbf{P_i^C} = \begin{bmatrix} P_{i_x}^C & P_{i_y}^C & P_{i_z}^C \end{bmatrix}$,

is projected to a 2D point on the image plane, $\mathbf{p_i} = \begin{bmatrix} p_{i_u} & p_{i_v} \end{bmatrix}$, as shown in Figure 2.2. The equation that relates the two points is:

$$\begin{aligned} p_{i_u} &= -\frac{fP_{i_x}^C}{P_{i_z}^C} \\ p_{i_v} &= -\frac{fP_{i_y}^C}{P_{i_z}^C} \end{aligned} \tag{2.5}$$

where f is the focal length of the camera and can be obtained through camera calibration.

Substituting from Equation 2.4 gives:

$$\begin{aligned} p_{i_u} &= -\frac{f(\mathbf{R_i}^T\mathbf{P_i^O} + T_x)}{\mathbf{R_k}^T\mathbf{P_i^O} + T_z} \\ p_{i_v} &= -\frac{f(\mathbf{R_j}^T\mathbf{P_i^O} + T_y)}{\mathbf{R_k}^T\mathbf{P_i^O} + T_z} \end{aligned} \tag{2.6}$$

This transformation is non-linear, adding another layer of difficulty to the problem.

2.3 Feature Selection

To further explore the methods of solving the pose estimation problem, one must be able to model the target object. The object is assumed known a priori. It is usually described by a set of features, and a feature is any descriptor of a shape.

For example, a common feature is a point. An object can be described with a set of points which indicates the location of the corners found in that object. The triangle in Figure 2.3 can be described by the set {(0,0,1), (0,1,0), (1,0,0)}. A point can also represents the centroid of a hole or perhaps an apex of a curve. Another feature that is commonly used is a line, which is used to represent an edge of the object.

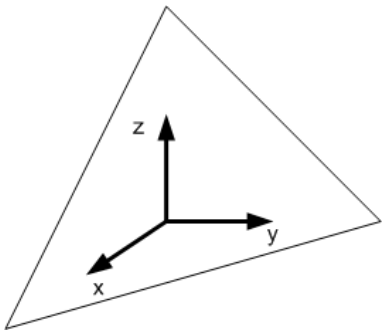
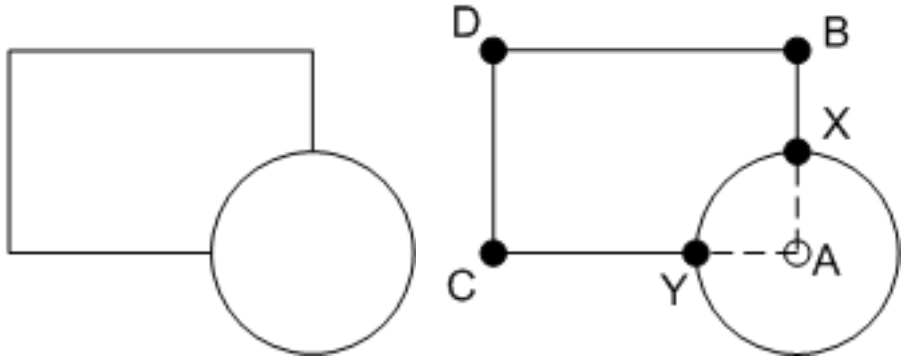


Figure 2.3: A triangle in 3D space

For a feature to be useful, it must be possible for the feature to be reliably measured from one or multiple images. A point feature used to represent a corner

can be measured using a corner detection algorithm such as the curvature scale space corner detector[14]. A line can be measured using an edge detector, such as Canny[3], and combined with Hough transform[11].

In a realistic scenario, the image is not perfect. Some of the features from the model may be blocked from the view of the camera, and this is known as occlusion. The feature detection algorithm may detect features that are not part of the model, and these extra spurious features are called clutter. The image will always be affected by noise and the image processing algorithm may be reliable. All these factors will contribute to the selection of features used for pose estimation.



(a) A rectangle occluded by a circle (b) Opaque points are likely to be detected

Figure 2.4: A situation where using lines may be more advantageous to using points as features

Comparing points with lines, point features have an advantage in terms of speed. Point features are simpler than lines and allows operations such as rotation and translation to be performed in a shorter time. However, points are easily occluded and spurious points may generated when occlusion occurs. Figure 2.4 is used to illustrate this point. Figure 2.4(a) shows a scenario where one corner of a rectangle is occluded by a circle. Suppose the rectangle is an image of the target model. The opaque points in 2.4(b) are points that are likely to be detected. Notice X and Y would be detected instead of A and those two points do not correspond to any features in the rectangle. If a line is used, the line BX and CY will be detected, which can be used to match the corresponding line in the model. The performance of the two feature classes for the algorithm of interest in this thesis will be compared in this thesis.

2.4 Pose Estimation with Correspondence

To properly formulate the problem, consider the situation where there are 2 sets of points of size m: a 3D set representing the model denoted by $\mathbf{P^O} = \begin{bmatrix} \mathbf{P_1^O} & \mathbf{P_2^O} & \cdots & \mathbf{P_m^O} \end{bmatrix}$,

and a 2D set detected from the image denoted by $\mathbf{p'} = \begin{bmatrix}\mathbf{p'_1} & \mathbf{p'_2} & \cdots & \mathbf{p'_m}\end{bmatrix}$. Assume it is known that each point, $\mathbf{p'_i}$, in the 2D set is the image of $\mathbf{P_i^O}$ in the 3D set.

Let the 3D point set, $\mathbf{P^C}$, be the points $\mathbf{P^O}$ with respect to the camera frame instead of the model frame. Thus $\mathbf{P^C}$ and $\mathbf{P^O}$ are related by Equation 2.3. The rotation, $\mathbf{R}$, and the translation, $\mathbf{T}$ are unknown. Let $\mathbf{p} = \begin{bmatrix}\mathbf{p_1} & \mathbf{p_2} & \cdots & \mathbf{p_m}\end{bmatrix}$ be the projection of the $\mathbf{P^C}$ onto the image plane. Thus, points between $\mathbf{P^C}$ and $\mathbf{p}$ are related by Equation 2.5.

Figure 2.5 shows an image of the cube and the projection of the model. The grey cube is part of the image and the detected points would be denoted as $\mathbf{p'}$. The wire-frame is the projection of the model of a given pose. Since the pose may not be accurate, the wire-frame and the image may not overlap. The features on the wire-frame is denoted by $\mathbf{p}$.

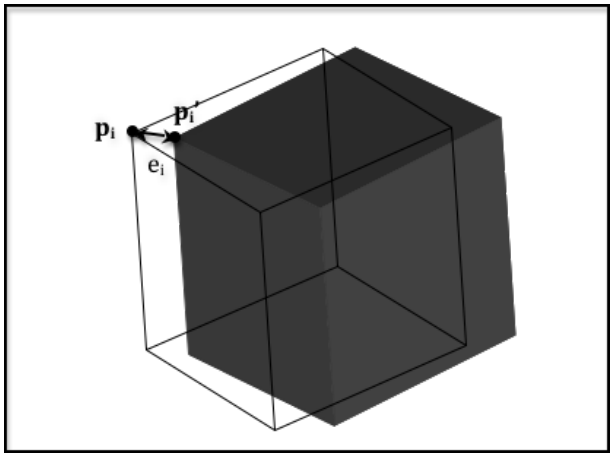


Figure 2.5: Image of a cube with the projection of the model

Now, an error vector, $\mathbf{E} = \begin{bmatrix}e_1 & e_2 & \cdots & e_m\end{bmatrix}$, can be defined between $\mathbf{p}$ and $\mathbf{p'}$. The error may be any distance measure between the two points and the square of the Euclidean distance is commonly used. This would give:

$$e_i = ||\mathbf{p_i} - \mathbf{p'_i}||^2 \tag{2.7}$$

The pose estimation problem is now reduced to finding $\mathbf{R}$ and $\mathbf{T}$ such that $||\mathbf{E}||$ is minimized, where $\mathbf{R}$ satisfies the constraints in Equation 2.1.

2.5 The Correspondence Problem

Unfortunately, the correspondence between the features is usually not known. When features are detected from the image, there is usually no indication whether a detected feature is created by the object or which feature of the object it belongs to. This leads to a closely related to the pose estimation problem known as the

correspondence problem. It is the process of finding out which features in a set correspond to a feature in another set.

If the pose of the target is known, the correspondence problem becomes trivial. One can simply project the model onto the image plane and associate each projected model feature to the closest image feature to determine the correspondence. On the other hand, pose estimation is also much simpler if the correspondence is known as it becomes much simpler to judge whether a pose is good or not. In the scenario where both feature sets are in the 3D space, pose estimation with correspondence can be solved with a closed form solution using quarternion[16] and orthogonal matrices [17][32]. For the purpose of visual servoing, the model feature set is in 3D while image feature set is in 2D. This problem is more complicated than a 3D-to-3D or a 2D-to-2D case due to the non-linearity introduced by the perspective transformation of the camera.

The problem of pose estimation without correspondence is also known as the simultaneous pose and correspondence problem. This is the focus of this thesis.

2.6 Related Work

The simultaneous pose estimation and correspondence problem has been the focus of a number of researchers throughout the years. One motivation to solve this problem is to allow for greater flexibility for the initialization of feature-based visual servoing. Once the initial pose is found, subsequent pose can be tracked using Extended Kalman Filter as shown by Wilson et. al. [33].

The pose estimation problem is a much simpler problem if the correspondence is known. As mentioned previously, a 3D-to-3D or 2D-to-2D case can be solved using quarternion[16] or orthogonal matrices [17][32]. In a 3D-to-2D scenario, the perspective transformation needs to be considered. Fishcler and Bolles[12] coined the term Perspective-n-Point problem (or PnP problem) when there is n number of points. The closed form solutions for the P3P problem have been formulated by Dementhon et. al.[9] and Fishcler et. al.[12]. The P3P problem is important as it is the lowest value of n which has a limited number of solutions. For a higher number of n, an iterative approach is proposed by Lowe [19] using Gauss-Newton Method and by Dementhon [10] with the POSIT method. Both of these methods will be further discuss in this thesis.

Of course, the correspondence between features are usually unknown. A number of researchers only consider the 3D-to-3D or 2D-to-2D scenarios. Some of these researchers focus on the spatial relationship and attempt to find the pose without introducing the concept of correspondence. Barrow et al.[1] introduced chamfer matching to minimize the distance between line features. Hong and Tan[15] proposed the use of a canonical form which allows for an affine transformation on the object rather than just translations and rotations. The canonical form proposed by Hong and Tan is a type of hash function for objects in the 2D plane. It allow

the pose to be calculated very efficient in $O(n)$ time, but is unable to handle cases where the number of features in the model is different from the number of features in the image.

Other researchers uses the spatial relationship between the features as the constraints to the problem and focus on finding the correspondence. Scott and Longuet-Higgins[27] uses singular value decomposition to find the correspondence by minimizing distance between two point sets. Their method does not handle rotation well. Shapiro and Brady[28] improved upon their method to handle rotation better by considering the intra-spatial relation between two point sets instead of the inter-spatial relation within the set. However, it still does not function well when the number of features between model and the image greatly differs.

There are also methods which consider both the spatial relationship and the correspondence. Gold et. al.[13] proposed the SoftAssign method which alternatively solves for the pose and the correspondence. Chui and Rangarajan[6] proposed the robust point matching algorithm which allows for non-rigid object. The iterative closest point algorithm[2] and its variants[26] are often used in 3D-to-3D pose estimation. These methods are similar in the way they attempt to solve for the pose and the correspondence individually. They solve the correspondence by assuming certain pose and solve the pose from the correspondence. They mainly differs in the way the correspondence is solved.

All above mentioned methods only work when both the image features set and the model feature set are in the same dimensional space, but studying those methods may give insight on solving the 3D-to-2D pose estimation problem. Some 3D-to-2D pose estimation methods are based on methods that only works for feature sets in the same dimensional space.

One approach to the 3D-to-2D pose estimation and correspondence problem is the hypothesis-and-test approach, which assumes certain correspondence between a small subset of the data. A pose is calculated from this correspondence, and then the pose is tested for correctness. The RANSAC method[12] is a well-known example of such approach and its effectiveness will be examined in this thesis. Lowe[20] proposed the use of scale invariant feature transform (SIFT) to allow for easier correspondence, but only performs well for models with complex texture.

Dementhon created the SoftPOSIT method[8] which is based on their original POSIT method[10] and the SoftAssign method proposed by Gold et. al.[13]. The SoftAssign method is used for solving pose estimation problem when both sets of data are in the same dimensional space. It alternates between solving the pose estimation problem and the correspondence problem. The SoftPOSIT method is based on the same concept, but incorporates the POSIT method such that the algorithm works for data sets with different dimensions.

Previous researchers mostly focused on whether a method is able to find a good pose or not without much regards to any time constraints. This thesis will consider the performance of various methods under a 30 second time limit. The limit is arbitrary and it simply a time that is considered reasonable for the initialization

for a real time application. The SoftPOSIT will be the focus of this research. This method require an initial guess of the pose. Each initial pose may or may not lead to the correct pose. However, if the correct pose is not found, the algorithm can be ran again on a different pose. This thesis aims to improve the running time by detecting cases where the algorithm may terminate early to restart on a different pose.

Chapter 3

Pose Estimation With Correspondence

The simultaneous pose and correspondence problem is difficult to tackle. One approach to divide the problem and solve for the pose and the correspondence separately. The idea is to first assume an initial pose and solve for the correspondence. Since a pose is assumed, solving for the correspondence becomes relatively simple. The newly found correspondence will then be used to find a new pose. The assumption is that this new pose will be an improvement over the previous one because a better correspondence is used. The process is repeated with the hope that the pose will improve at each iteration until a good one is found. While there is no guarantee that the correct pose will be achieved, there is a tendency for the algorithm to find a better pose at each iteration. If the correct pose is not found, the algorithm can simply be restarted with a different initial pose.

Essentially, by using this method, the simultaneous pose estimation and correspondence problem is separated into two simpler problems. This chapter will discuss the Gauss-Newton method[19] and the POSIT method[10] of solving pose estimation when correspondence is given.

3.1 Gauss-Newton Method

Since the goal is to minimize the cost function in Equation 2.7, a logical choice of algorithm is to use a general optimization technique for non-linear functions. One such method is the Gauss-Newton Method (GNM) suggested by Lowe[19]. GNM is a method for solving non-linear least square problems. The general outline of the algorithm is described below, and how it applies to the pose estimation problem is described later in this section.

Given m nonlinear residual functions, $r_1, r_2, \ldots, r_m$ that depend upon n parameters, $\beta = \begin{bmatrix}\beta_1 & \beta_2 & \ldots & \beta_n\end{bmatrix}$, the goal of GNM is to find β such that the sum of the square of the residual, $\sum_{i=1}^{m} r_i^2$, is minimum. In the case of pose estimation, the

residual function, r_i, can be the distance between a projected model feature and a measured feature in the image. The parameters β will represent the pose.

The vector β is set to an initial value and updated based on the residual function and the Jacobian matrix, $\mathbf{J}$

$$J_{ij} = \frac{\partial r_i}{\partial \beta_j} \tag{3.1}$$

The GNM iterates by updating the parameter vector, β, with the equation

$$\beta^{i+1} = \beta^i - \Delta, \tag{3.2}$$

where β^i is the value of β in the i^{th} iteration and Δ is the parameter update vector which minimizes the 2-norm of the residual

$$||\mathbf{J}\Delta - \mathbf{r}||^2 \tag{3.3}$$

GNM can be derived from the Taylor's theorem, which states:

$$\mathbf{r}(\beta^{i+1}) \approx \mathbf{r}(\beta^i) - \mathbf{J}\Delta \tag{3.4}$$

Therefore, by minimizing $||\mathbf{r}(\beta^i) - \mathbf{J}\Delta||^2$, $||\mathbf{r}(\beta^{i+1})||^2$ is approximated to minimised. From 3.3, we have:

$$\begin{aligned} ||\mathbf{J}\Delta - \mathbf{r}||^2 &= (\mathbf{J}\Delta - \mathbf{r})^T(\mathbf{J}\Delta - \mathbf{r}) \\ &= \Delta^T\mathbf{J}^T\mathbf{J}\Delta - \Delta^T\mathbf{J}^T\mathbf{r} - \mathbf{r}^T\mathbf{J}\Delta + \mathbf{r}^T\mathbf{r} \\ &= \Delta^T\mathbf{J}^T\mathbf{J}\Delta - 2\Delta^T\mathbf{J}^T\mathbf{r}^T + \mathbf{r}^T\mathbf{r} \end{aligned} \tag{3.5}$$

Differentiating with respect to Δ gives:

$$\frac{d||\mathbf{J}\Delta - \mathbf{r}||^2}{d\Delta} = 2\mathbf{J}^T\mathbf{J}\Delta - 2\mathbf{J}^T\mathbf{r} \tag{3.6}$$

Thus, the residual is minimised when:

$$\begin{aligned} 2\mathbf{J}^T\mathbf{J}\Delta - 2\mathbf{J}^T\mathbf{r} &= 0 \\ \mathbf{J}^T\mathbf{J}\Delta &= \mathbf{J}^T\mathbf{r} \end{aligned} \tag{3.7}$$

This gives the general outline of the algorithm. First, set the parameters, β, to an initial value β_0. Calculate the residual vector, $\mathbf{r}$ and the Jacobian, $\mathbf{J}$. Update β with Equation 3.2, where Δ satisfies Equation 3.7. β is updated iteratively until it converges.

To apply this method to pose estimation, suppose there is a 3D model point set, $\mathbf{P^O}$, and the corresponding 2D points, $\mathbf{p'}$, detected from an image. Let $\mathbf{P^C}$ be the transformation of $\mathbf{P^O}$ into the camera frame with rotation, $\mathbf{R}$, and translation, $\mathbf{T}$, where $\mathbf{R}$ and $\mathbf{T}$ represent the pose of the model. Let the 2D point set, $\mathbf{p} = \begin{bmatrix}\mathbf{p_1} & \mathbf{p_2} & \cdots & \mathbf{p_m}\end{bmatrix}$ be the projection of $\mathbf{P^C}$ onto the image plane. Then the residual, $\mathbf{r}$, can be defined as a $2m \times 1$ vector with

$$\begin{aligned} r_{2i-1} &= p_{i_u} - p'_{i_u} \\ r_{2i} &= p_{i_v} - p'_{i_v} \end{aligned} \tag{3.8}$$

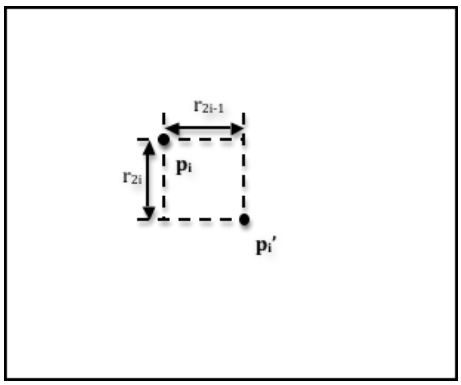


Figure 3.1: Illustration of the residual functions

The residuals in Equation 3.8 represent the horizontal and the vertical distance respectively between the image points i and the projected model points i. Figure 3.1 illustrate the residual functions.

To calculate the Jacobian, it is necessary to relate the derivatives between a 3D model point and its projection onto the image. Recall Equation 2.5 which is restated below for easier reference.

$$\begin{aligned} p_{i_u} &= -\frac{fP_{i_x}^C}{P_{i_z}^C} \\ p_{i_v} &= -\frac{fP_{i_y}^C}{P_{i_z}^C} \end{aligned} \tag{3.9}$$

Differentiating with respect to β gives

$$\begin{aligned} \frac{\partial p_{i_u}}{\partial \beta} &= -\frac{f}{P_{i_z}^C}\left(\frac{\partial P_{i_x}^C}{\partial \beta} - \frac{P_{i_x}^C}{P_{i_z}^C}\frac{\partial P_{i_z}^C}{\partial \beta}\right) \\ \frac{\partial p_{i_v}}{\partial \beta} &= -\frac{f}{P_{i_z}^C}\left(\frac{\partial P_{i_y}^C}{\partial \beta} - \frac{P_{i_y}^C}{P_{i_z}^C}\frac{\partial P_{i_z}^C}{\partial \beta}\right) \end{aligned} \tag{3.10}$$

Therefore, to determine the partial derivatives of a point in the image plane with respect to the parameters of transformation, one can simply apply the above equations on the partial derivatives of the point in 3D space. The parameters used will be the translation, $\mathbf{T} = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T$, and rotation matrix, $\mathbf{R}$.

Differentiating equations 2.4 with respect to $\mathbf{T}$ gives:

$$\frac{\partial (P_{i_x}^C, P_{i_y}^C, P_{i_z}^C)}{\partial \mathbf{T}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Substituting into Equation 3.10 gives

$$\frac{\partial(p_{i_u}, p_{i_v})}{\partial \mathbf{T}} = \begin{bmatrix} -\frac{f}{P_{i_z}^C} & 0 & \frac{fP_{i_x}^C}{P_{i_z}^{C\,2}} \\ 0 & -\frac{f}{P_{i_z}^C} & \frac{fP_{i_y}^C}{P_{i_z}^{C\,2}} \end{bmatrix} \tag{3.12}$$

Handling the rotation is more tricky. One method is to represent the rotation is using Euler angles, but this will make the calculation for the Jacobian difficult as rotation about one axis is dependent upon the rotation about another axis. To show this difficulty, suppose the roll-pitch-yaw system is used. $\mathbf{R}$ would be represented by $\mathbf{R} = \mathbf{R}_\alpha \mathbf{R}_\beta \mathbf{R}_\gamma$, where

$$\begin{aligned} \mathbf{R}_\gamma &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix} \\ \mathbf{R}_\beta &= \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \\ \mathbf{R}_\alpha &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.13}$$

The yaw(γ), pitch(β) and roll(α) will be used as the parameter, β. Explicitly multiplying the matrix gives:

$$\mathbf{R} = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha s_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \tag{3.14}$$

where s and c are abbreviation representing the sine and cosine function. Calculating the Jacobian involves differentiating combinations of the sine and cosine function, making it a time consuming process.

To simplify the computations, method of updating the rotation is modified from the typical definition used for GNM. Instead of subtracting a number to a certain parameter as in Equation 3.2, $\mathbf{R}$ will be updated by multiplying with 3 rotation update matrices, $\mathbf{R}_\phi$, $\mathbf{R}_\theta$ and $\mathbf{R}_\psi$, giving

$$\mathbf{R_{i+1}} = \mathbf{R}_\psi \mathbf{R}_\theta \mathbf{R}_\phi \mathbf{R_i} \tag{3.15}$$

where $\mathbf{R}_\phi$, $\mathbf{R}_\theta$ and $\mathbf{R}_\psi$ are the incremental update about the x, y, and z-axis re-

spectively in the counterclockwise direction. i.e.

$$\begin{aligned}
\mathbf{R}_\phi &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \\
\mathbf{R}_\theta &= \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \\
\mathbf{R}_\psi &= \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.16}$$

ϕ, θ and ψ are the amount of rotation that should be applied and is calculated using the Jacobian. Let $\bar{\mathbf{P}}_\mathbf{i} = \begin{bmatrix} \bar{P_{i_x}} & \bar{P_{i_y}} & \bar{P_{i_z}} \end{bmatrix}^T = \mathbf{R}\mathbf{P}_\mathbf{i}^\mathbf{O}$, then

$$\frac{\partial \bar{\mathbf{P}}}{\partial(\phi, \theta, \psi)} = \begin{bmatrix} 0 & -\bar{P_{i_z}} & \bar{P_{i_y}} \\ \bar{P_{i_z}} & 0 & -\bar{P_{i_x}} \\ -\bar{P_{i_y}} & \bar{P_{i_x}} & 0 \end{bmatrix} \tag{3.17}$$

Substituting into Equation 3.10 gives:

$$\frac{\partial(p_{i_u}, p_{i_v})}{\partial(\phi, \theta, \psi)} = \begin{bmatrix} \frac{f}{\bar{P_{i_z}}}\left(\frac{\bar{P_{i_x}}\bar{P_{i_y}}}{\bar{P_{i_z}}}\right) & -\frac{f}{\bar{P_{i_z}}}\left(\bar{P_{i_z}} + \frac{\bar{P_{i_x}}^2}{\bar{P_{i_z}}}\right) & \frac{f\bar{P_{i_y}}}{\bar{P_{i_z}}} \\ -\frac{f}{\bar{P_{i_z}}}\left(-\bar{P_{i_z}} - \frac{\bar{P_{i_y}}^2}{\bar{P_{i_z}}}\right) & -\frac{f}{\bar{P_{i_z}}}\left(\frac{\bar{P_{i_x}}\bar{P_{i_y}}}{\bar{P_{i_z}}}\right) & -\frac{f\bar{P_{i_x}}}{\bar{P_{i_z}}} \end{bmatrix} \tag{3.18}$$

Equations 3.12 and 3.18 allow the Jacobian to be calculated efficiently.

A problem with GNM is the lack of robustness of the method. The algorithm may not necessarily converge to a local minimum. In comparison, the steepest descent method has a higher chance of convergence, but the rate of convergence is slower than GNM. Thus, to improve the chance of convergence, Lowe[19] proposed the used of LevenbergMarquardt algorithm[23]. This method is essentially a combination of the GNM and the steepest descent method.

In this approach, a weighting matrix, W, is created. W is a diagonal matrix with elements, $w_{ii} = 1/\sigma_i$, where σ_i^2 is variance of parameter i. σ_i can be set based on prior knowledge of the system and can be set to 1 if the variance is unknown. This allows a higher weighting to the parameters with higher certainty. The update equation is changed to finding Δ such that the following is minimised:

$$(J^T J + \lambda W)\Delta = J^T r \tag{3.19}$$

λ is a variable to control the effect of the weighting matrix. It should be noted that if W is the identity matrix, a large λ will diminish the effect of $J^T J$ and Δ will be in the direction of steepest descent. Thus, this transforms into the steepest descent

algorithm which is more robust than GNM. Whereas if λ is small, the algorithm will behave more like GNM. λ is adjusted at each iteration of the GNM. Marquardt suggested to increase λ by a factor of 10 when divergence occurs and decrease λ by a factor of 10 when the algorithm is converging[22].

3.1.1 Alternative cost function

The GNM method minimised the sum of the square of the residual. Recall from Equation 3.8 that the described cost function is the distance measurement in the x and y direction and they are considered separately. A more intuitive method is to use one error metric for each point by changing the cost function. A cost function that can be used is the square of the Euclidean distance between two points. The residual function for image and object point i becomes:

$$r_i = ||\mathbf{p_i} - \mathbf{p'_i}||^2 \tag{3.20}$$

The Jacobian can be calculated with

$$J_{ij} = \frac{dr_i}{d\beta} = 2(p'_{i_u} - p_{i_u}) \times \frac{du}{d\beta} + 2(p'_{i_v} - p_{i_v}) \times \frac{dv}{d\beta} \tag{3.21}$$

Brief testing was performed where the translation is fixed and only the rotation is varied. Approximately 30% of the test cases did not converge properly when using the original cost function. Using the new cost function, all test cases converged. The tests performed was not extensive enough to draw concrete conclusions, but did indicate improvements in performance when the alternative cost function is used.

3.2 POSIT

The GNM is a generic algorithm for minimizing nonlinear cost functions. An algorithm that is designed specifically for pose estimation is like to produce better performance. One such algorithm is the POSIT algorithm proposed by Dementhon and Davis[10]. The POSIT algorithm stands for Pose from Orthographic Scaling with ITeration, and the method is explained below.

The transformation is defined by the 3x3 rotation matrix, $\mathbf{R}$, and the 3x1 translation vector $\mathbf{T} = \begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T$. Recall from Equation 2.2 that $\mathbf{R}$ can be expressed as

$$\mathbf{R} = \begin{bmatrix} \mathbf{R_i} & \mathbf{R_j} & \mathbf{R_k} \end{bmatrix}^T \tag{3.22}$$

where $\mathbf{R_i}$, $\mathbf{R_j}$ and $\mathbf{R_k}$ are the 3x1 camera coordinate vectors expressed in the object frame. The vectors satisfy the right-hand rule and thus, the equation

$$\mathbf{R_k} = \mathbf{R_i} \times \mathbf{R_j} \tag{3.23}$$

Therefore, it is only necessary to determine $\mathbf{R_i}$ and $\mathbf{R_j}$ to be able to calculate the full rotation matrix.

3.2.1 Scale Orthographic Projection (SOP)

The POSIT algorithm retrieves the pose by calculating the scaled orthographic projection (SOP) of the model points. The SOP is a model which assumes the distance between the model and the camera is significantly greater than the size of the model. This is not necessarily true in the application. The image points that are retrieved in reality is the perspective projection of the object. However, if it is possible to determine the SOP of the model, then the pose can be determined easily as shown in this section. Figure 3.2 is used to illustrate this type of projection.

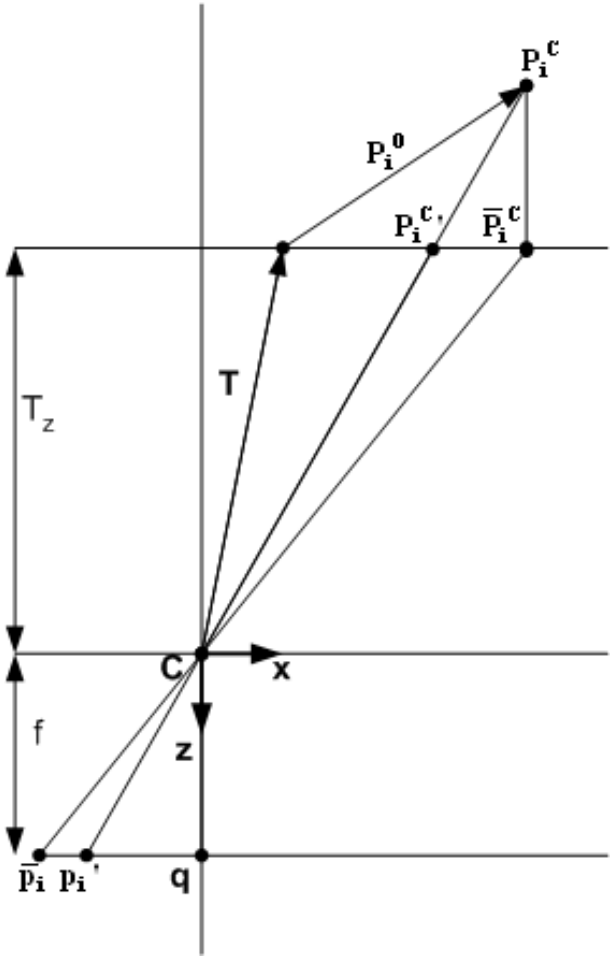


Figure 3.2: Geometric representation of the SOP

Suppose given a 3D point of the model, $\mathbf{P_i^C} = \begin{bmatrix} P_{i_x}^C & P_{i_y}^C & P_{i_z}^C \end{bmatrix}^T$, of the model with respect to the camera frame, projective transformation transforms $\mathbf{P_i^C}$ into the point $\mathbf{p_i}$ on the image plane with

$$p_{i_u} = -\frac{f}{P_{i_z}^C} \cdot P_{i_x}^C, \qquad p_{i_v} = -\frac{f}{P_{i_z}^C} \cdot P_{i_y}^C \tag{3.24}$$

With SOP, all model points are assumed to have the same distance in the z direction

from the camera. It is instead transformed into the point $\overline{\mathbf{p_i}} = \begin{bmatrix}\overline{p_{i_u}} & \overline{p_{i_v}}\end{bmatrix}^T$ with

$$\overline{p_{i_u}} = -\frac{f}{T_z} \cdot P^C_{i_x}, \qquad \overline{p_{i_v}} = -\frac{f}{T_z} \cdot P^C_{i_y} \tag{3.25}$$

Note that the translation vector, $\mathbf{T}$, is also the location of the origin of the model in the camera frame. Let $s = -\frac{f}{T_z}$ and substitute from Equation 2.4, gives

$$\begin{aligned} \overline{p_{i_u}} &= s(\mathbf{R_i}^T \mathbf{P_i^O} + T_x) \\ &= s(\mathbf{P_i^O}^T \mathbf{R_i} + T_x) \\ &= \begin{bmatrix}\mathbf{P_i^O}^T & 1\end{bmatrix} \begin{bmatrix} s\mathbf{R_i} \\ sT_x \end{bmatrix} \end{aligned} \tag{3.26}$$

Similarly,

$$\overline{p_{i_v}} = \begin{bmatrix}\mathbf{P_i^O}^T & 1\end{bmatrix} \begin{bmatrix} s\mathbf{R_j} \\ sT_y \end{bmatrix} \tag{3.27}$$

These two equations form a linear system with $s\mathbf{R_i}$, sT_x, $s\mathbf{R_j}$ and sT_y as unknowns. Each image point gives two equations. Therefore, if 4 or more SOP points are given, the problem becomes over constrained, and can be solved with standard least square minimization technique. Since $\mathbf{R_i}$ and $\mathbf{R_j}$ are unit vectors, they can be determined from

$$\mathbf{R_i} = \frac{s\mathbf{R_i}}{||s\mathbf{R_i}||} \tag{3.28}$$

$$\mathbf{R_j} = \frac{s\mathbf{R_j}}{||s\mathbf{R_j}||} \tag{3.29}$$

s is simply the magnitude of $s\mathbf{R_i}$ and $s\mathbf{R_j}$. Thus

$$s = ||s\mathbf{R_i}|| \tag{3.30}$$

sT_x and sT_y are solved from the linear system. Therefore, solving for s allows T_x, T_y to be found. T_z can also be calculated as $s = -\frac{f}{T_z}$ and $\mathbf{R_k} = \mathbf{R_i} \times \mathbf{R_j}$ from Equation 3.23. Thus, the full transformation is recovered.

3.2.2 Perspective transformation

Knowing the SOP of the model points allows the pose to be recovered. However, the image of the model provides the perspective projection of the model points instead. To solve for pose from a true perspective projection, it is necessary to calculate the locations of the SOP points as follows.

Theorem. The SOP of a model point, $\overline{p_i}$, and the perspective project of the same point, p'_i, is related by the equation:

$$\overline{p_{i_u}} = {p_{i_u}}'(1+\epsilon_i) = \begin{bmatrix}\mathbf{P_i^O}^T & 1\end{bmatrix} \begin{bmatrix} s\mathbf{R_i} \\ sT_x \end{bmatrix} \tag{3.31}$$

$$\overline{p_{i_v}} = {p_{i_v}}'(1+\epsilon_i) = \begin{bmatrix}\mathbf{P_i^O}^T & 1\end{bmatrix} \begin{bmatrix} s\mathbf{R_j} \\ sT_y \end{bmatrix} \tag{3.32}$$

where

$$\epsilon_i = \frac{1}{T_z}\mathbf{R_k}^T\mathbf{P_i^O} \tag{3.33}$$

Proof. Consider Figure 3.2 with **C** being the origin of the camera. $\overline{\mathbf{p_i}}$ is the SOP of $\mathbf{P_i^C}$ and p'_i is the perspective projection of $\mathbf{P_i^C}$. Consider only the x direction, $\overline{p_u}$ can be expressed as

$$\overline{p_u} = p'_{i_u} + (\overline{p_u} - p'_{i_u}) \tag{3.34}$$

Since $\overline{p_u}$ and p'_{i_u} are perspective projection of $\bar{P}_{i_x}^C$ and $P_{i_x}^{C'}$, they are related by a scaling factor of f/T_z, giving

$$\begin{aligned}(\overline{p_u} - p'_{i_u}) &= \frac{f}{T_z}(\bar{P}_{i_x}^C - P_{i_x}^{C'}) \\ &= \frac{f}{T_z}(P_{i_x}^C - P_{i_x}^{C'})\end{aligned} \tag{3.35}$$

Due to similar triangle $\mathbf{Cqp'_i}$ and $\mathbf{P_i^{C'}\bar{P}_i^C P_i^C}$,

$$(P_{i_x}^C - P_{i_x}^{C'}) = \frac{P_{i_z}^C - T_z}{f}p'_{i_u} \tag{3.36}$$

Substituting from Equation 2.4 gives

$$(P_{i_x}^C - P_{i_x}^{C'}) = \frac{\mathbf{R_k}^T\mathbf{P_i^O}}{f}p'_{i_u} \tag{3.37}$$

Combining equations 3.34, 3.35 and 3.37 gives

$$\begin{aligned}\overline{p_u} &= p'_{i_u} + \frac{1}{T_z}\mathbf{R_k}^T\mathbf{P_i^O}p'_{i_u} \\ &= p'_{i_u}(1+\epsilon), \text{where } \epsilon = \frac{1}{T_z}\mathbf{R_k}^T\mathbf{P_i^O}\end{aligned} \tag{3.38}$$

Substituting 3.38 into 3.26 gives 3.32. Similar procedures can be followed for the v direction to give 3.32. Equation 3.38 implies that the SOP of a point can be calculated if ϵ_i is known. Given the SOP, it is then possible to calculate the pose from 3.26 and 3.27.

□

3.2.3 The POSIT algorithm

Assuming the value of ϵ_i for all i is known, the pose can be retrieved by solving a set of equations from 3.32. Initially, ϵ_i is not known. It can be assumed to be zero to retrieve an estimate of the pose. With this pose estimate, it will then be possible to calculate a more accurate value of ϵ_i. The equations can be solved iteratively to achieve progressively more accurate data. The outline of the algorithm is shown below.

1. Initialize ϵ_i to 0 for $i = 1, 2, \cdots, m$.
2. Solve for $s\mathbf{R_i}$, sT_x, $s\mathbf{R_j}$ and sT_z from Equation 3.32.
3. Calculate s by normalizing $s\mathbf{R_i}$ to solve for T_x, T_y and T_z.
4. Calculate $\mathbf{R_k} = \mathbf{R_i} \times \mathbf{R_j}$.
5. Update ϵ_i with Equation 3.33
6. Repeat from 2 until convergence.

3.3 Comparison between Gauss-Newtown Method and POSIT

The algorithms are tested using Monte Carlo simulation, which essentially test the performance of the algorithms using randomly generated scenario[24]. 50 sets of 5 to 15 randomly generated 3D points are created to represent the model. For each set, a random translation and rotation is created as the pose of the model. The translation is bounded such that it is within the view of the camera.

The point sets are transformed and projected to create 2D point sets. The GNM and the POSIT algorithms are applied to the 2D and 3D sets to recover the transformation, and a successful match is one where the Euclidean distance between each image feature and the projection of model feature is below a threshold. This threshold depends on the tolerance level of the visual servoing application that the pose estimation is used for.

For the GNM, an initial transformation is required and the effect of this transformation is also tested. The initial translation will be set to the actual translation of the model. This is usually not possible in practice without additional sensors, which implies the GNM will be able to perform better in this test than it would in real world environment. For the rotation, a random axis of rotation is chosen. 36 rotation matrices are created using the chosen axis with magnitude of rotation ranging from 5° and 180° in 5° intervals. This rotation is applied to the actual rotation of the model and used as the initial transformation. Thus, for each point set, 36 initial transformations will be applied and tested.

The test is performed on GNM and POSIT. Figure 3.3 shows the success rate of the GNM method against the error in the initial rotation. The POSIT method does not require an initial pose estimate, therefore, the data is not shown on the graph. Testing have shown that the POSIT method is able to achieve retrieve a correct pose for all 50 sets of points. On the other hand, the performance of GNM degrades as the initial rotation differs from the correct rotation.

The GNM fails often when it is unable to converge to a pose. Perhaps this can be solved by using alternative stabilization method. However, this test has already

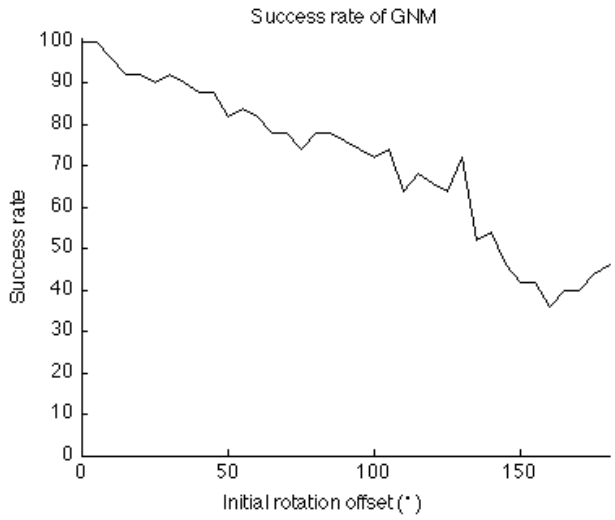


Figure 3.3: Success rate for Gauss-Newton method

proven the POSIT to be a much more reliable method. Thus, the final algorithm should be based on POSIT rather than GNM and it was decided not to invest more time and effort on GNM.

Chapter 4

Simultaneous Pose and Correspondence Problem

The previous chapter discussed the methods for solving the pose estimation with correspondence problem. In reality, the correspondence is usually not given. Thus, a method to solve for the pose without correspondence is required. This is also known as the simultaneous pose and correspondence problem. A few methods are presented in this chapter.

4.1 SoftPOSIT Method with Point Features

Gold et. al. [13] has proposed the SoftAssign method for point matching. Their method, however, only works with 2D-to-2D or 3D-to-3D point matching. Dementhon et. al.[8] combines the idea behind the SoftAssign method and POSIT method to all for 3D-to-2D matching. This section describes the SoftPOSIT method as proposed by Dementhon.

Softassign separates the pose estimation and the correspondence problem. The correspondence between the feature can be represented by a match matrix. The match matrix, $\mathbf{M}$, is a zero-one matrix, where $M_{ij} = 1$ implies a correspondence between image feature i and model feature j. It must satisfy the constraints that each entry is either zero or one, and the sum of any rows or columns must be also be one. The correspondence problem involves finding this match matrix.

Consider Figure 4.1 as an example. On the left of the figure is a model of the pyramid with its features labelled from 1 to 4. On the right, the is an image of the pyramid. The corners of the image is also labelled from 1 to 4, but the order is different from the model. The correspondence matrix, $\mathbf{M}$, for this figure would be

$$\mathbf{M} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

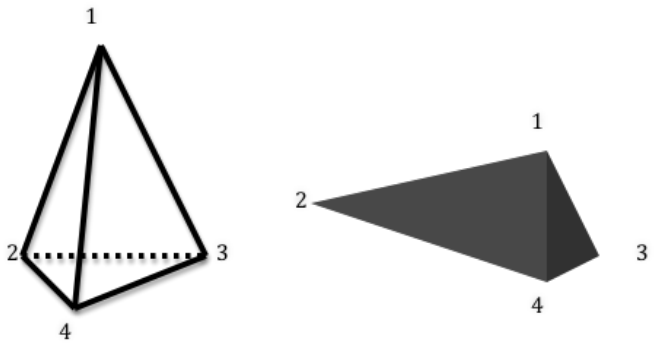


Figure 4.1: Model of a pyramid and its image

Now consider the case where there are n image points and m model points. Then the match matrix, M, will be n-by-m matrix. Enforcing the permutation matrix implies the problem is a discrete problem. By relaxing the constraint, it is possible to transform this problem into a continuous problem, which is easier to solve as many optimization methods rely on the data being continuous. Thus, only the doubly stochastic constraints will be used. A doubly stochastic matrix is one where the sum of rows and columns has to equal to 1, but the entries in the matrix can be in the interval $[0, 1]$ instead of strictly one or zero. A high value for M_{ij} indicates a strong match between model feature j and image feature i. A slack row and column can be added, transforming M into and n+1-by-m+1 matrix. The slack rows and columns allow for the possibility of features not having a match. With a slack column, the correspondence of image feature i that does not belong to the model can be represented by setting the value in the slack column of row i to 1 and the rest of the entries in that row 0. The constraints can be expressed as

$$\begin{aligned} M_{ij} \in [0,1] \quad &, \quad \text{for } 1 \leq i \leq n+1 \text{ and } 1 \leq j \leq m+1, \\ \sum_{k=1}^{m+1} M_{ik} \leq 1 \quad &, \quad \text{for } 1 \leq i \leq n, \\ \sum_{k=1}^{n+1} M_{kj} \leq 1 \quad &, \quad \text{for } 1 \leq j \leq m \end{aligned} \tag{4.2}$$

The algorithm begins by assuming a random initial pose represented by $\mathbf{R_0}$ for the rotation and $\mathbf{T_0}$ for the translation. With this pose, a projection of the model can be made. A matrix, $\mathbf{d}$, is calculated, where d_{ij} is the distance between image feature i and model feature k. This matrix, $\mathbf{d}$, indicates the proximity of a model feature to an image feature given the initial pose. A low value for entry d_{ij} implies model feature i is close to image feature j, which would suggest those two features are a good match.

The match matrix, $\mathbf{M}$, is calculated from a matrix $\mathbf{Q}$, where $\mathbf{Q} = \exp(-\beta \mathbf{d})$. β is a parameter that starts at a low value and increases for each iteration by

multiplying with a constant, δ. A high value in $\mathbf{Q}$ indicates a strong match between two features. There are a couple of reasons for creating $\mathbf{Q}$ with the exponent. The first reason is that it makes all the entries positive, which is a necessary condition to apply the Sinkhorn's method described later in this section. The second reason is that as β increases, the exponent increases the difference between two pairs of matches when the distance is different. At the beginning of the algorithm when β is low, entries $\mathbf{Q}$ will be close to 1 even if the distance between two features are large. This means a model feature will be equally matched with a number of image features. As β increases, the values in $\mathbf{Q}$ will decrease rapidly unless d_{ij} is close to 0. Therefore, as the algorithm progresses, a model feature will have to be very close to an image feature for it to have any significance towards the pose estimation.

As the matrix $\mathbf{Q}$ is an indicator of how well a pair of feature matches, a high value of Q_{ij} should result in a high value of M_{ij}. However, a feature might have similar distance to a number of other features. This is the reason why the binary constraints for $\mathbf{M}$ is relaxed. The strength of how well a pair of features are matched can be indicated by a value in the interval $[0, 1]$ instead of a binary value of 0 and 1. The doubly stochastic constraints needs to be maintained such that a feature does not have higher significance in the pose estimation stage simply because it has a closer proximity to more image features. As a result, $\mathbf{M}$ should be a doubly stochastic matrix that maximises the cost function, $E(M)$, where

$$E(M) = \sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij} Q_{ij} \tag{4.3}$$

This can be performed with the Sinkhorn's method[30] as proposed by Sinkhorn. The method repeatedly normalizes the rows and the columns a matrix and is proven to converge. This method has the following form:

1. Set $\mathbf{M^0}$ to $\mathbf{Q_{ij}}$
2. Update $\mathbf{M}$ by normalizing across all rows:
   $M_{ij}^{k+1} \leftarrow \frac{M_{ij}^{k}}{\sum_{j=1}^{m+1} M_{ij}^{k}}$, for $i = 1, 2, \cdots, n$
3. Update M by normalizing across all columns
   $M_{ij}^{k+1} \leftarrow \frac{M_{ij}^{k}}{\sum_{i=1}^{n+1} M_{ij}^{k}}$, for $j = 1, 2, \cdots, m$
4. Repeat from 2 until $\mathbf{M}$ converges. i.e. until $||\mathbf{M^k} - \mathbf{M^{k-1}}||$ is small.

The match matrix provides the weighting to use when solving for the pose using the POSIT method. Recall that a linear system can be created from Equation 3.32. For the POSIT method, the linear system is solved using standard least square minimization method. For the SoftPOSIT method, the match matrix is use

to assign a weighting for each equation in the system. Therefore, this becomes a weighted least square minimization problem. The solution to the system is

$$\begin{bmatrix} s\mathbf{R_i} \\ sT_x \end{bmatrix} = \left(\sum_{j=1}^{m} Mj' \mathbf{P_j^O} \mathbf{P_j^{O}}^T \right)^{-1} \left(\sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij}(1+\epsilon_j) p'_{ju} \mathbf{P_j^O} \right)$$

$$\begin{bmatrix} s\mathbf{R_j} \\ sT_y \end{bmatrix} = \left(\sum_{j=1}^{m} Mj' \mathbf{P_j^O} \mathbf{P_j^{O}}^T \right)^{-1} \left(\sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij}(1+\epsilon_j) p'_{jv} \mathbf{P_j^O} \right) \tag{4.4}$$

$$\tag{4.5}$$

where $M'_j = \sum_{i=1}^{n} M_{ij}$. Appendix A provides more details about the weighted least square minimization problem.

Obtaining the match matrix allows a new pose to be calculated using the Soft-POSIT method, which in turns will update $\mathbf{Q}$. Given a new $\mathbf{Q}$, the Sinkhorn's method can be used to obtain a new $\mathbf{M}$. β is increased in each iteration. The process is repeated until a satisfactory result is found or β reaches its final value.

The entire algorithm is presented below:

1. Initialize $\beta \leftarrow \beta_0$, $\mathbf{T} \leftarrow \mathbf{T_0}$, $\mathbf{R} \leftarrow \mathbf{R_0}$,
2. Apply transformation of $\mathbf{R}$ and $\mathbf{T}$ to the model and project model points onto the image plane
3. Compute $\mathbf{d}$ and $\mathbf{Q} = \exp(-\beta \mathbf{d})$
4. Apply the Sinkhorn's method
   (a) Set M_{ij}^0 to Q_{ij}
   (b) Update $\mathbf{M}$ by normalizing across all rows: $M_{ij}^{k+1} \leftarrow \frac{M_{ij}^k}{\sum_{j=1}^{m} M_{ij}^k}$
   (c) Update $\mathbf{M}$ by normalizing across all columns: $M_{ij}^{k+1} \leftarrow \frac{M_{ij}^k}{\sum_{i=1}^{n} M_{ij}^k}$
   (d) Repeat from (b) until $\mathbf{M}$ converges
5. Solve for $s\mathbf{R_i}$, sT_x, $s\mathbf{R_j}$ and sT_z from Equation 4.5
6. Calculate s by normalizing $s\mathbf{R_i}$ to solve for T_x, T_y and T_z
7. Calculate $\mathbf{R_k} = \mathbf{R_i} \times \mathbf{R_j}$
8. Update ϵ_i with Equation 3.33
9. Update $\beta \leftarrow \beta \times \delta\beta$
10. Repeat from 2 until $\beta \geq \beta_f$

The constants used in the algorithm are found through testing. It depends on the magnitudes of the values obtained during for the distance measurement. A high value of β indicates the confidence in the correctness of the pose. It should start at a low value, but should be high enough such that the entries in $\mathbf{Q}$ are not all close to 1 whether or not the features are close to each other. If the starting value is too low, then the initial pose will become meaningless and most of the initial poses will result in the same final pose when the algorithm converge. On the other hand, if the starting value is too high, the algorithm will often run into numeric problems as the values in the $\mathbf{Q}$ matrix approaches 0. The final value of β should be high enough to allow the algorithm to converge. Through trial-and-error, the values used in this thesis is $\beta_0 = 0.01$, $\beta_f = 0.1$, and $\delta = 1.05$.

4.2 SoftPOSIT Method with Line Features

Line features have the advantage that it is more robust to partial occlusion when compared with points. David et. al. propose a method to extend the SoftPOSIT algorithm to handle line features instead of point features[7]. The idea is to convert the lines into points and apply the SoftPOSIT method on the points obtained. The method is described below.

As proposed by David et. al.[7], a line can be defined by two end points. Suppose a 2D line in the image plane, l_i, is define by the points $\mathbf{p_i}$ and $\tilde{\mathbf{p}}_\mathbf{i}$. This line creates a plane that passes through l_i and the camera center $\mathbf{C}$ (see Figure 4.2). The normal of the plane is $\mathbf{n_i} = (\mathbf{p_i}, f) \times (\tilde{\mathbf{p}}_\mathbf{i}, f)$. Any point, $\mathbf{P}$ on this plane will satisfy the equation $\mathbf{n_i}^T\mathbf{P} = 0$.

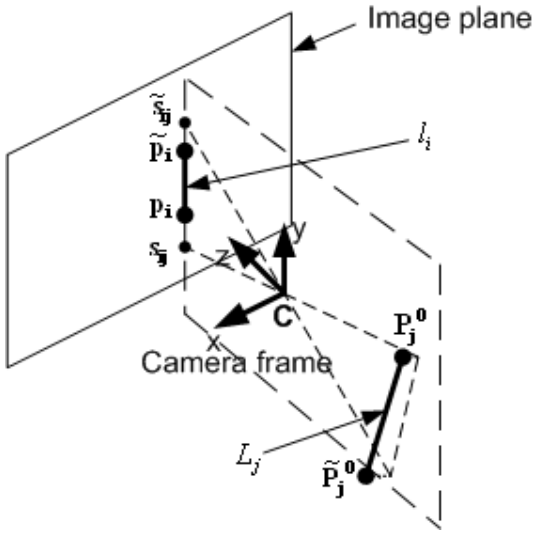


Figure 4.2: Projection of a line

Suppose this 2D line correspond to the 3D line $L_j^O = \{\mathbf{P_j^O}, \tilde{\mathbf{P}}_\mathbf{j}^\mathbf{O}\}$. The object has a pose given $\mathbf{R}$ and $\mathbf{T}$. The points $\mathbf{P_j^O}$ and $\tilde{\mathbf{P}}_\mathbf{j}^\mathbf{O}$ will be transformed to $\mathbf{P_j^C} =$

$\mathbf{RP_j^O} + \mathbf{T}$ and $\mathbf{\tilde{P}_j^C} = \mathbf{R\tilde{P}_j^O} + \mathbf{T}$. If $\mathbf{R}$ and $\mathbf{T}$ are perfect, then $\mathbf{P_j^C}$ and $\mathbf{\tilde{P}_j^C}$ will lie on the plane. Otherwise, there will be certain distance between 3D points and the plane, with the distance equals $\mathbf{{n_i}^T P}$. Let the points, $\mathbf{S_{ij}}$ and $\mathbf{\tilde{S}_{ij}}$ be points on the plane i closest to P_j^C and $\tilde{P_j^C}$. This gives the equations

$$\mathbf{S_{ij}} = \mathbf{P_j^C} - (\mathbf{{n_i}^T P_j^C})\mathbf{n_i}, \tag{4.6}$$

$$\mathbf{\tilde{S}_{ij}} = \mathbf{\tilde{P}_j^C} - (\mathbf{{n_i}^T \tilde{P}_j^C})\mathbf{n_i} \tag{4.7}$$

By projecting, $\mathbf{S_{ij}}$ and $\mathbf{\tilde{S}_{ij}}$ onto the image plane, we get $\mathbf{s_{ij}}$ and $\mathbf{\tilde{s}_{ij}}$, where

$$\mathbf{s_{ij}} = \frac{(S_{ij_x}, S_{ij_y})}{S_{ij_z}}, \mathbf{\tilde{s}_{ij}} = \frac{(\tilde{S_{ij_x}}, \tilde{S_{ij_y}})}{\tilde{S_{ij_z}}} \tag{4.8}$$

Each 3D point pair, $\mathbf{P_j^O}$ and $\mathbf{\tilde{P}_j^O}$, is matched with every 2D line, l_i, to produce $\mathbf{s_{ij}}$ and $\mathbf{\tilde{s}_{ij}}$. For every 3D lines, there are two endpoints. Thus, if there are m 3D model lines and n detected 2D image lines, there will be $2m$ 3D object points and a $2nm$ 2D image points.

The SoftPOSIT method can be used to perform pose estimation on the point sets instead of the line sets. The match matrix, $\mathbf{M}$, will be calculated from the distance between the lines instead of the endpoints, and M_{ij} will be used to represent the match strength between the endpoints of line L_j to line l_i.

The distance measure should incorporate both the difference in the orientations between two lines and the separation between the lines. The difference in the orientation between l_i and l_j is measured by $\Delta\theta(l_i, l_j) = 1 - |\cos \angle l_i l_j|$, where $\angle l_i l_j$ is the angle between the two lines. Notice that maximum difference in the angle is $-\pi/2$ as reflected by the measurement function. The separation between two lines, $d(l_i, l_j)$, is defined as the sum of the distance of the endpoints of the projected line, l_j, to the closest point on the image line, l_i. The distance, d_{ij}, measure between two lines can then be defined as a linear combination of the two measurements:

$$d_{ij} = k\Delta\theta(l_i, l_j) + d(l_i, l_j) \tag{4.9}$$

where k adjust the relative importance between the orientation and the separation. However, testings have shown that the incorporation of the difference in the angle is not necessary. Consider Figure 4.3 where the solid line is the image and the dotted line is the projection of the model line. Simply considering the distance between the endpoints is sufficient to represent this error. A difference in the angle between two lines lead will lead a difference in the separation, making the consideration on the orientation redundant. Equation 4.9 is included for reference, but for this thesis, k is set to 0 to reduce the amount of calculation.

4.3 Termination Condition

There is only a fraction of possible initial poses that will lead to a good pose at convergence. When the SoftPOSIT method converges, it should determine whether

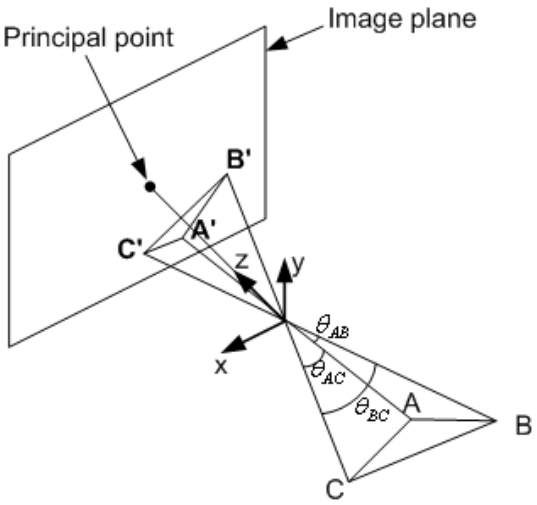


Figure 4.3: Demonstration of the error between two lines

a good pose is obtained. If not, the algorithm should restart with a different initial pose.

A good pose can be determined by the number of feature matches the algorithm is able to obtain. A pair of features is considered a match if it satisfies two conditions. The first condition is that the distance between the two features is within a threshold, σ. This threshold is dependent on the application and represents the maximum allowable error for each feature. The second condition involves examining the match matrix, $\mathbf{M}$. A high value in the match matrix indicates two features are strongly matched to each other. If the entry, M_{ij}, is highest in the row, it implies the image feature i is better matched to model feature k than any other model features. Likewise, if M_{ij} is highest in column k, it implies model feature k is best matched to image feature i. Therefore, an entry that is highest in both its row and column indicate that pair of features is best match with each other, and this is the second condition for a pair to be considered as a match.

The algorithm should terminate when the number of matches equals to or exceeds the number of features captured in the image. Suppose p_d is the percentage of features from the model detected in the image, then the number of features detected will equal $p_d m$. In reality, the value of p_d is usually not known, but can often be estimated with some a prior information on the target model and the scene for pose estimation. Therefore, the algorithm can terminate when the number of match equals or exceeds t_m, which is defined as

$$t_m = \rho p_d m \tag{4.10}$$

where $0 < \rho < 1$. The purpose of ρ is to account for some amount of errors in feature detection and the uncertainty in p_d. It lowers the requirement for a pose to be considered as a good pose. Dementhon et al.[8] suggested the value of 0.8 for ρ.

This termination condition is not perfect and may result in a good pose to be

considered as a bad one or vice versa, but it is fairly reliable if the number of model features present in the scene is relatively high.

For testing purpose, the value of p_d is assumed to be known. For real application, this value should be estimated.

4.4 Early Search Termination

A problem with the SoftPOSIT method is that the method iterates over a range of value for β whether the algorithm converges to a good pose or not. However, it is often possible for a human being to determine whether the algorithm will converge to a good pose within a few iteration. If the algorithm is able to make this judgement, then improvement upon the runtime performance can be expected.

The pose estimation of the model should improve upon each iteration, and distance between two corresponding features should decrease over time. It is noted that if the algorithm is not converging towards a good pose, the distance tends to decrease slower. This leads to the proposed solution.

The idea is to find the number of model features that is less than a certain distance, $\bar{\sigma}$, an image feature. If the number is less than $t_m = \rho p_d m$, terminate the algorithm early and restart with a different initial pose. Notice that this is basically a relaxed version of the termination condition mentioned in the previous section.

The value of $\bar{\sigma}$ should be sufficiently high in the beginning to prevent the algorithm from terminating early every time it starts. This should decreases over each iteration and the final value of $\bar{\sigma}$ should equal to the threshold, σ, discussed in the previous section. The SoftPOSIT algorithm tends to converge faster at the beginning and slower at the end. Therefore, $\bar{\sigma}$ should have have this property. With these conditions in mind, $\bar{\sigma}$ is proposed to be inversely proportional to the control parameter of the SoftPOSIT algorithm, β. Thus,

$$\bar{\sigma} = \frac{\sigma \beta_f}{\beta} \tag{4.11}$$

where β_f is the final value of β. However, this equation does not allow the initial value to be adjusted. In the test scenarios performed in section 5.3, this was not an issue for this thesis and the early termination condition is able to improve the SoftPOSIT algorithm. If it is desirable to adjust the initial value and the rate of decrease of $\bar{\sigma}$, the following equation is proposed:

$$\bar{\sigma} = \alpha(\frac{\sigma \beta_f}{\beta} - \sigma) + \sigma \tag{4.12}$$

where α is a control parameter. Equation 4.12 allows the initial value to be adjusted with α. It keeps the property that $\bar{sigma}$ decrease faster in the beginning and have the final value at $\bar{sigma} = \sigma$. The value of α is dependent on the application. A

higher value of α increases the initial value of $\bar{\sigma}$, which decreases the chance of the algorithm terminating when it would have eventually converge to a good pose. A low value of α increases the aggressiveness of the early termination condition, allowing for more bad pose to be terminated early. Setting α to 1 will give 4.11.

To summarize, the outline of the early search termination is as follows:

1. Project model onto the image with current pose estimate.
2. For each projected model feature, k, find the closest image feature and calculate the distance between the two features and store in matrix, $\mathbf{D}$
3. Set $\bar{\sigma} = \alpha(\frac{\sigma\beta_f}{\beta} - \sigma) + \sigma$
4. Count the number of elements in $\mathbf{D}$ that are smaller than $\bar{\sigma}$
5. If the number is less than $\rho p_d m$, restart SoftPOSIT with a different initial pose. Otherwise continue with the SoftPOSIT algorithm.

4.5 RANSAC

An alternative method for pose estimation is the RANdom SAmple Consensus (RANSAC) method [12]. The idea of the RANSAC method is to start with the smallest set of correspondence. In the case of pose estimation, the correspondence of 3 non-collinear points are required. With this initial matching, a pose is estimated. Any features that are greater than a certain threshold distance from the model are rejected as outliers. If the number of features close to the model exceeds a certain number, those features will be use to produce a more refined pose estimation.

Given a number of matching points, the possible locations and the orientations of the camera can be calculated. The problem of finding the camera position based on matched image points is known as the Location Determination Problem(LDP). If the principal point and the focal length of the camera are known, it is possible to calculate the angle between the lines connecting any pair of image points connecting to the camera (see Figure 4.4). Therefore an alternate statement of the LDP is:

> Given a model containing n points and the angle to every pair of points from the camera location, find the lengths of line segment joining the camera location to each model point. This is known as the 'perspective-n-point' problem (PnP).

The RANSAC method uses the minimum number of points required to initiate the pose estimation. The P1P ($n = 1$) and the P2P ($n = 2$) problems have an infinite number of solutions. The P3P problem does not give unique solution, but the number of solutions is limited to a maximum of 4. Therefore, the RANSAC

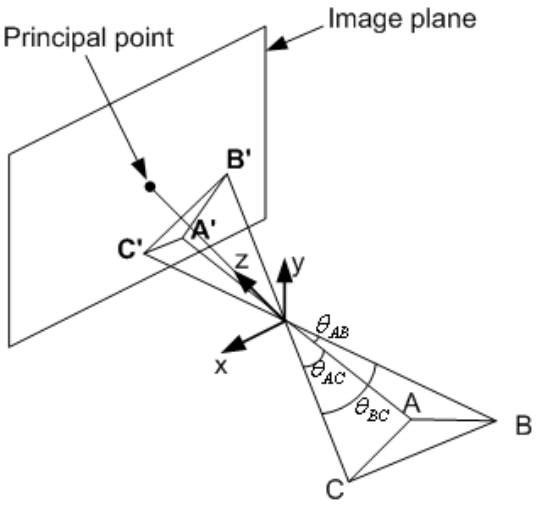


Figure 4.4: Geometry of Location Determination Problem

method uses 3 points for correspondence. The P3P ($n = 3$) requires the determination of 3 edges of a tetrahedron given the dimension of the base and the face angle of the opposing corner. The solution to the P3P problem is shown in appendix B. Readers can also refer to Fishcler[12] for details.

The outline of the RANSAC algorithm is shown below:

1. Randomly select 3 image points and 3 model points.
2. Solve the P3P problem as described in appendix A to get a pose estimation
3. Apply pose estimation to the model and calculate the distance of each model point to the closest point in the image.
4. Consider all points with distance within certain threshold as inliers and use the POSIT method to solve for a pose.
5. Repeat until a good pose is found based on the termination condition described in section 4.3.

4.6 Simulated Annealing

Since pose estimation can be treated as an optimization problem, it is logical to assume a global optimization technique may be able to provide a suitable pose estimation. Simulated annealing (SA) is one such technique and was chosen to test this approach. This approach was explored by Starnik and Backer[31]. It attempts to escape from local minimum by allowing the error of a solution to increase from the previous solution with a probabilistic chance. The idea of simulated annealing is an analogy of physical annealing process used for metal treatment.

The SA approach focuses on finding the correspondence instead of the pose. A solution is represented by a list, $\mathbf{s}$, of size m, where m is the number of features in the model. The entry, s_i of the list represent the matching assigned to model feature i. $s_i = j$ means that model feature i is matched with image feature j. A value of 0 indicates the model feature is not matched to any features in the image. A new neighbour solution, $\mathbf{s}_{new}$ is created by one of the following:

1. Swap the matches of two features by swapping two entries in $\mathbf{s}$.
2. Remove a match for a feature by setting an entry in $\mathbf{s}$ to 0.
3. Create a match for an unmatched features by setting a zero entry in $\mathbf{s}$ to a non-zero value that is not already in $\mathbf{s}$.

It should be ensured that none of the model features are matched to the same image feature.

SA is controlled by the temperature parameter, T. The value of T is set to an initial value of T_0 at the beginning and updated according to the cooling schedule. A random solution is created at the start of the algorithm and the error, E, is calculated with the equation:

$$E = \sum_{i=1}^{m} e_i \tag{4.13}$$

where

$$e_i = \begin{cases} d_{max}, & \text{if } l_i = 0 \\ d(i, l_i), & \text{otherwise} \end{cases} \tag{4.14}$$

$d(i, l_i)$ is the distance measure between model feature i and image feature l_i. d_{max} is the maximum distance where a match is considered acceptable. A match with distance higher than d_{max} means it is better for the model feature to be unmatched instead.

At each step of SA, a nearby solution is tested. If the solution has a lower error than the previous one, the new solution is accepted and replaces the old one. If the solution has a higher error, then there is a probability that this solution is still accepted. The probability depends on the difference in the error between the new and the old solution and T. It has the form:

$$P = \exp\left(\frac{(E_{new} - E)}{T}\right) \tag{4.15}$$

where E is the original error, E_{new} is the error of the new solution. A higher difference in error and lower temperature will decrease the probability of accepting a solution.

At each value of T, N solutions are created and tested. N is determined through trial-and-error, and $N = 30$ was found to provide good performance for this thesis.

The acceptance rate, α, is the fraction of solutions that are accepted. After N solutions are tested, the temperature is modified based on the cooling schedule. T will decrease, which implies the probability of accepting a solution decreases over time. An adaptive cooling schedule is employed. The idea of this adaptive approach is that if many of the solutions are accepted, then the algorithm is most likely just accepting many random solutions. If few solutions are accepted, the algorithm is stuck in a local minimum. Therefore, the cooling of the temperature is set in attempt to keep the acceptance rate at a certain level. If the acceptance rate is too high, the temperature will drop rapidly. Whereas if the acceptance rate is too low, the temperature will decrease slowly. At each iteration, the temperature is updated to:

$$T_{i+1} = \begin{cases} T_i\delta_1, \text{if } \alpha > 0.5 \\ T_i\delta_2, \text{if } \alpha \leq 0.5 \end{cases} \tag{4.16}$$

The value of δ_1 and δ_2 is to be tweaked based on testing with $1 < \delta_1 < \delta_2 < 0$. The value of $\delta_1 = 0.9$, $\delta_2 = 0.97$ and $T_0 = 100$ was found to provide the good performance for this thesis. However, it should be noted that extensive testing was not performed to find the optimal value.

The procedure of SA is as follows:

1. Initialize $T \leftarrow T_0$
2. Create random initial solution, $\mathbf{s}$
3. Find neighbour solution of $\mathbf{s}_new$ and replace $\mathbf{s}$ with $\mathbf{s}_{new}$
4. Calculate a pose based on the correspondence and calculated an error, E_{new}, from the projected model
5. If new error is less than or equal previous error, accept new solution
6. If new error is greater than previous error, accept the solution with probability $P = \exp(\frac{(E_{new}-E)}{T})$
7. Repeat from step 3, N number of times
8. Calculate acceptance rate for the current temperature
9. Update temperature based on acceptance rate
10. Determine if the solution is good using the termination condition described in previous section
11. Repeat from step 3 until a good solution is found or $T \leq T_f$

The three methods described (SoftPOSIT, RANSAC and simulated annealing) are tested against each other and the results are shown in the next chapter.

Chapter 5

Results

Three solutions for simultaneous pose and correspondence problem are described in the previous chapter. It is necessary to test their performances in solving the pose estimation problem. A Monte Carlo simulation is used for most tests. There are a few issues that need to be examined. The first test compares the performance between using lines as features and points as were examined. Then, the effectiveness of the early termination condition for the SoftPOSIT method is tested. A comparison is drawn between the SoftPOSIT method, RANSAC and SA.

This thesis also examine the feasibility of the solutions in a more realistic environment. CAD models are produced to see if the SoftPOSIT method is able to retrieve a good pose from an image of the models. An example of the SoftPOSIT algorithm being used on a photograph is also given.

The testing method and results are given in this chapter.

5.1 Testing Method

The three algorithms described in the previous chapter are tested using Monte Carlo simulation. The simulations are based on a number of parameters and they are similar to the simulations used by Dementhon et al. [8] for their evaluation of the SoftPOSIT method.

The parameters are N, m, p_d, p_c. N is the number of trials performed for each combination of the other parameters. m is the number of features in the model. The features may be lines or points. p_d is the percentage of model features detected in the image. This is to simulate the possibility of occlusion and the fact that an image processing algorithm may not be able to detect all features in the image. p_c is the percentage of feature in the image does not belong to the 3D model. This parameter simulate additional features, or clutter, that may be found in a real life environment.

The simulation is created by assuming a lab environment where the object size is less than 0.2m $\times$ 0.2m $\times$ 0.2m and the camera has a focal length of 4mm,

or 4×10^{-3}m. The object size is approximately the size of a human hand and such focal length is found in typical low speed camera that one might use for visual servoing. For each simulation, a 3D model is generated by placing random uniformly distributed features in a $0.2 \times 0.2 \times 0.2$ unit area. A random translation and rotation is generated and applied to the model. The translation is constrained such that the model stays within the view of the camera. The transformed model is projected onto an 2D plane using a focal length of 4×10^{-3} unit to generated the image. A number of features will be removed from image and/or extra features added based on the parameter p_d and p_c respectively. An algorithm will be executed with the 3D model features and 2D image features as input to determine the pose of the transformed model, and the result will be evaluated to determine if the algorithm has executed successfully.

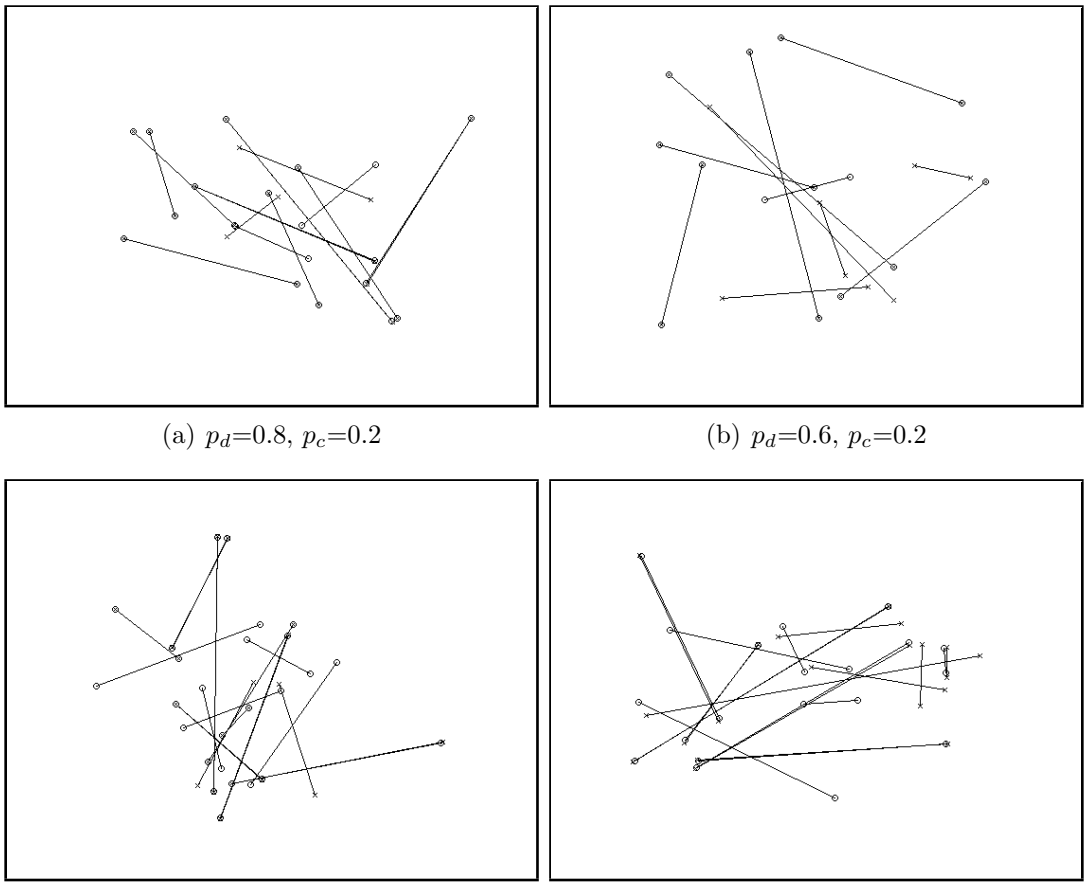
To determine if the transformation obtained is successful, the 3D model is transformed with the pose retrieved and projected using the same focal length to generated another image. The two images are compared. The transformation is good if each feature of the 3D model that is present in the original image is within a certain threshold distance of the same feature in the newly obtained image. Figure 5.1 and 5.2 display some examples of the test images with successful matches. These two figures provide a visual illustration on how the value of p_d and p_c affects the image. In both figures, a circle marker represents the image feature and a cross represents the projection of the model features. Slight errors are acceptable as visual servoing applicable usually have a small tolerance on the initial pose of the object.

A time limit is imposed on the evaluation. Each test is allowed to run for 30 seconds on an Intel Core 2 duo 2.4 GHz processor. 30 seconds is considered to be a reasonable time as a part of the initialization of a real time application. The algorithms may terminate before the 30 seconds time limit if it determines that a good pose is found, but if 30 seconds have passed and a good pose is not yet found, the algorithm is forced to stop and the test is considered as a failure.

5.2 Line vs Points

The choice of feature should be dependent on a number of factors. One of factors is if the chosen feature can be reliably extracted from an image and the accuracy of the measurement of the extracted feature. While this is an important factor, the focus of this research is the pose estimation after features are obtained and the image processing aspect will not be discussed in detail.

Even ignoring the image processing aspect, the choice of feature still has an impact on the performance of an algorithm as shown in this section. A point feature is simple and allows for fast calculations. When used with the SoftPOSIT method, each iteration is executed in a shorter time. Naturally, one may assume points features will perform better than lines if the properties of feature detection is ignored.



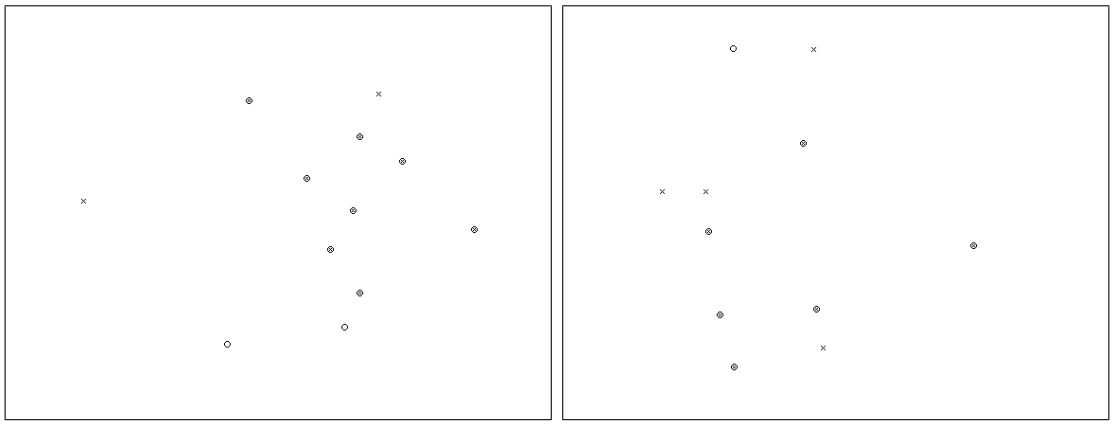
(a) p_d=0.8, p_c=0.2 (b) p_d=0.6, p_c=0.2

(c) p_d=0.8, p_c=0.4 (d) p_d=0.6, p_c=0.4

Figure 5.1: Samples of successful pose estimation with lines as features

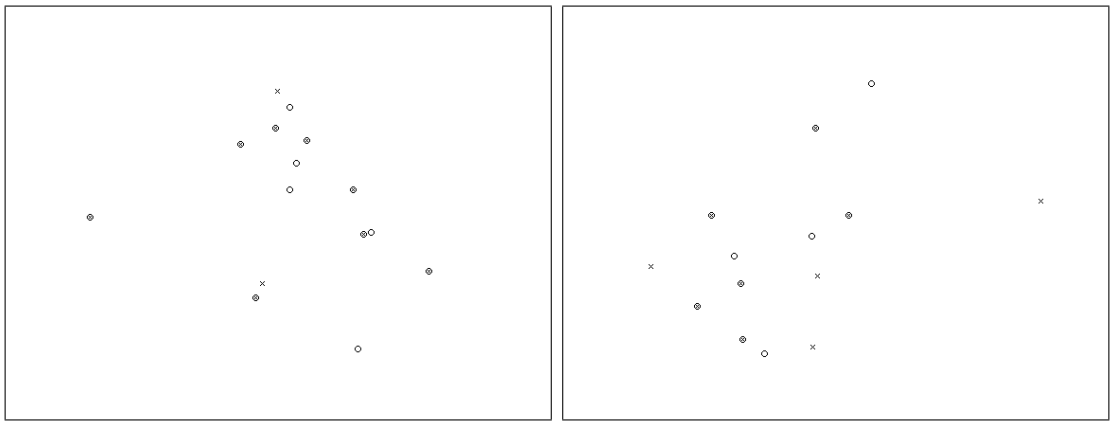
However, it is found that this is not the case in reality. A point feature contains less information than a line, making it more difficult to differentiate between two different points. Figure 5.3 is used to illustrate this point. Figure 5.3(a) is an image showing the projection of the model in a given pose onto the image. In such a situation, one may easily associate each of the model points with a unique image point and guess that the pose will eventually converge. However, when lines are added to connect the points, it can be seen that the pose estimate is obviously different from the correct pose of the model.

Another reason that lines may perform better is that lines allow the algorithm to converge faster. Consider the situation in Figure 5.4 where triangle ABC should be matched with triangle XYZ. The distance between point A to point X is the same as the distance between distance to point Y. Both point X and Y will try to "pull" point A towards itself. Eventually, the triangle will move towards the left due to the other points, but it will require a few iterations before point A is strongly matched with point X. On the other hand, due to the difference in the orientation, line AC would be considered a much better to line XZ than line XY or line ZY. Therefore, line AC will be strongly matched with XY right from the



(a) p_d=0.8, p_c=0.2

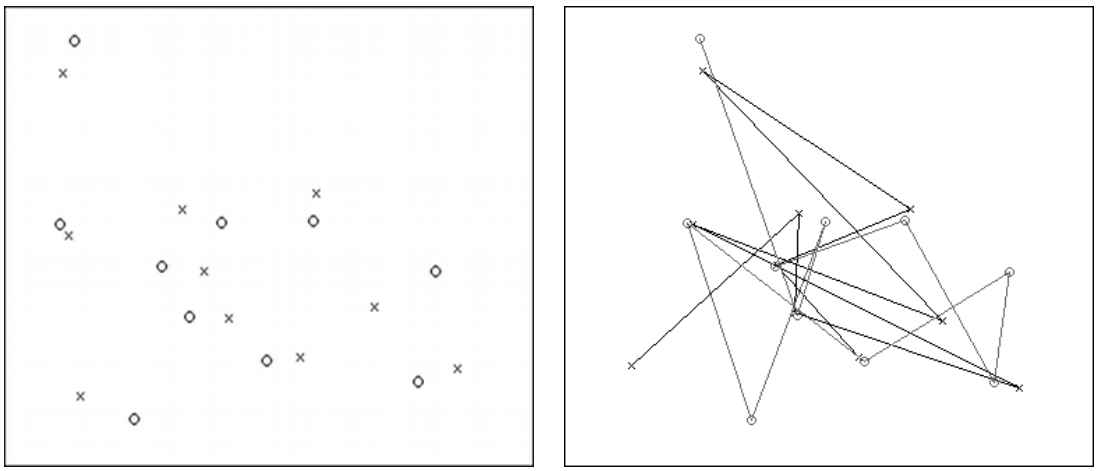
(b) p_d=0.6, p_c=0.2



(c) p_d=0.8, p_c=0.4

(d) p_d=0.6, p_c=0.4

Figure 5.2: Sample of successful pose estimation with points as features



(a) The image points are marked by ∘, projection of the model points are marked by x

(b) Same projection with lines connecting the points

Figure 5.3: Visual comparison between using points as features and lines as features

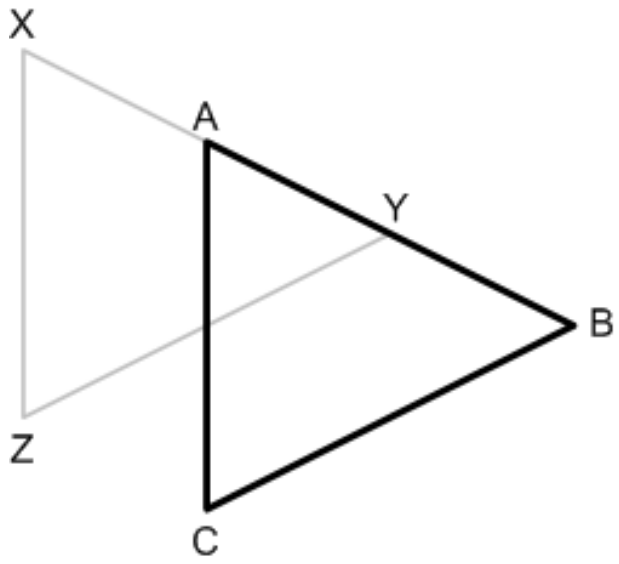


Figure 5.4: Attempt to match triangle ABC to triangle XYZ

beginning, allow the algorithm to converge with fewer iterations. Such situations often occur during pose estimation. Since all the points are very similar, a model point is often weakly matched with multiple image points. The extra information in a line often allows a model line to be strongly matched with fewer image lines.

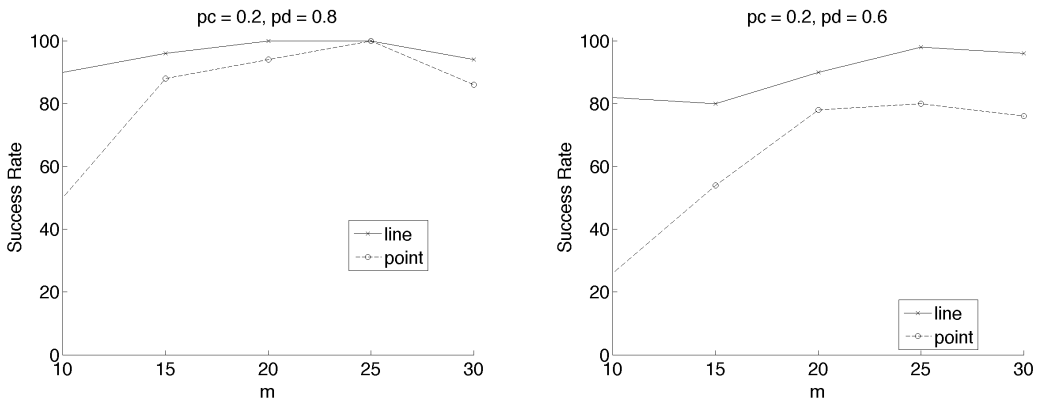
Test results also supports this theory. When using points as features, the SoftPOSIT method has to run more iterations for it to converge (i.e. β_f has to be increased). Otherwise, the termination condition does not perform well as the algorithm is still converging when the condition is reached, which result in a lot of false positives.

Figure 5.5 shows the difference in success rate between using points and lines. A lot of the failures when using points are the result of finding a false positive for the pose. This is especially significant when the number of points is low. Increasing clutter or occlusion also affects points more than lines. The improvement of using lines can be seen under all situations tested.

5.3 Effectiveness of Early Termination Condition

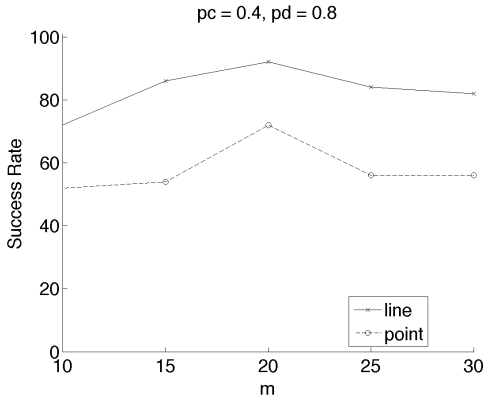
To test the effectiveness of the early termination condition, the Monte Carlo method described in previous section is used. The clutter level and the percentage of occlusion is set in each test, and the number of lines are varied.

The early termination condition should be able to detect failing cases earlier and restart the algorithm with a new initial pose. Therefore, the number of initial poses tested should increase. Figure 5.6 compares the difference in the number of initial poses between the SoftPOSIT method with and without the early termination

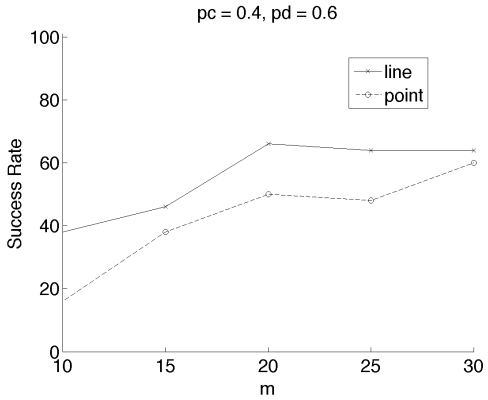


(a) $p_c = 0.2$, $p_d = 0.8$

(b) $p_c = 0.2$, $p_d = 0.6$



(c) $p_c = 0.4$, $p_d = 0.8$



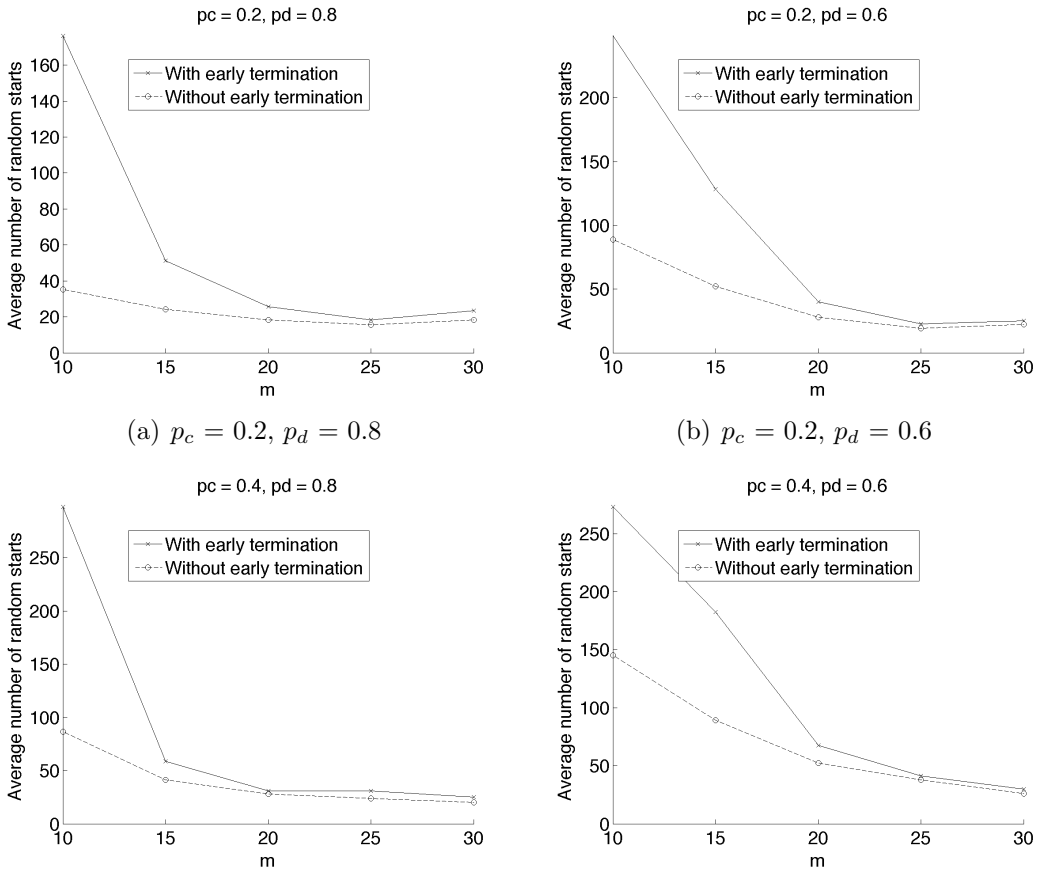
(d) $p_c = 0.4$, $p_d = 0.6$

Figure 5.5: Comparison of success rate between using lines and points as feature

condition. For a low number of features, it can be seen that the number of initial poses tested increased by as much as 5 times with early termination. It may be more appropriate to consider the number of random starts only for the failed cases. When the algorithm is able to find the correct pose quickly, the early termination condition would allow the algorithm to execute faster, but would not increase the number of random starts. Figure 5.7 shows the number of random starts only for the failed cases. This is essentially the number of initial poses that are tested within 30 seconds. It can be seen the early termination conditions allow the SoftPOSIT method to test approximately 30-50% more poses. The missing data in 5.7(a) is caused by the test cases having a 100% success rate.

As a result, the early termination condition allows SoftPOSIT method to increase the success rate slightly. The increase in the number of tested pose allowed approximately 5% more cases to become successful. The difference between success rate is shown in Figure 5.8.

A more significant impact is the time requirement of the algorithm. The early termination of decreases the amount of time spent in exploring failed cases. This decreases the running time of the algorithm by as much as 50%. A detail comparison



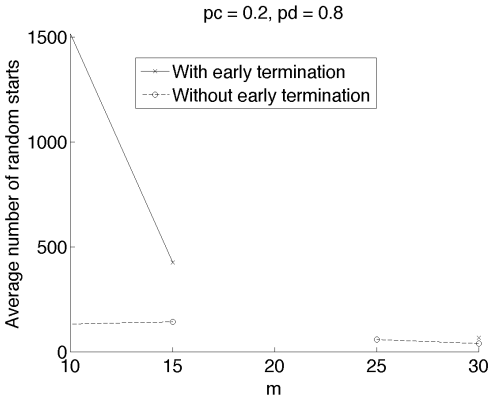
(a) $p_c = 0.2$, $p_d = 0.8$

(b) $p_c = 0.2$, $p_d = 0.6$

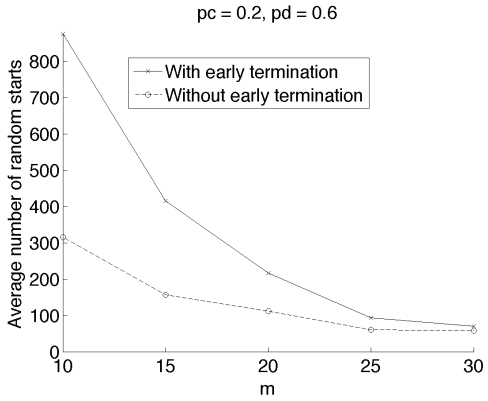
(c) $p_c = 0.4$, $p_d = 0.8$

(d) $p_c = 0.4$, $p_d = 0.6$

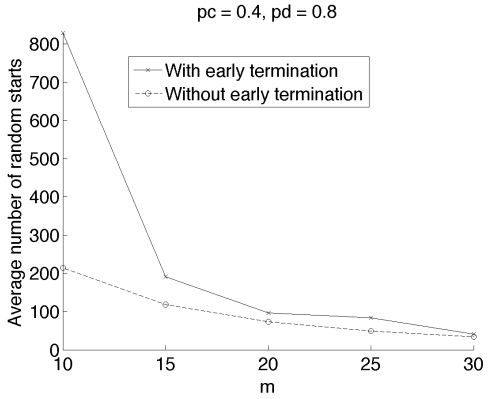
Figure 5.6: Comparison of the number of random starts with and without early termination



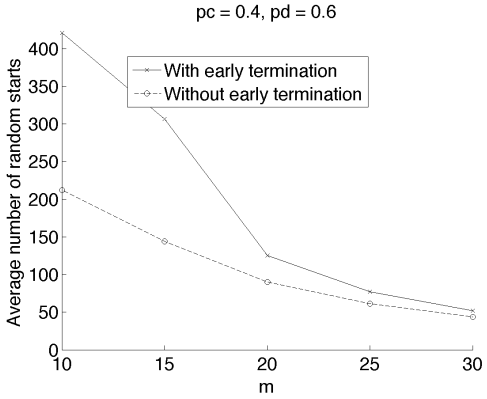
(a) $p_c = 0.2$, $p_d = 0.8$



(b) $p_c = 0.2$, $p_d = 0.6$

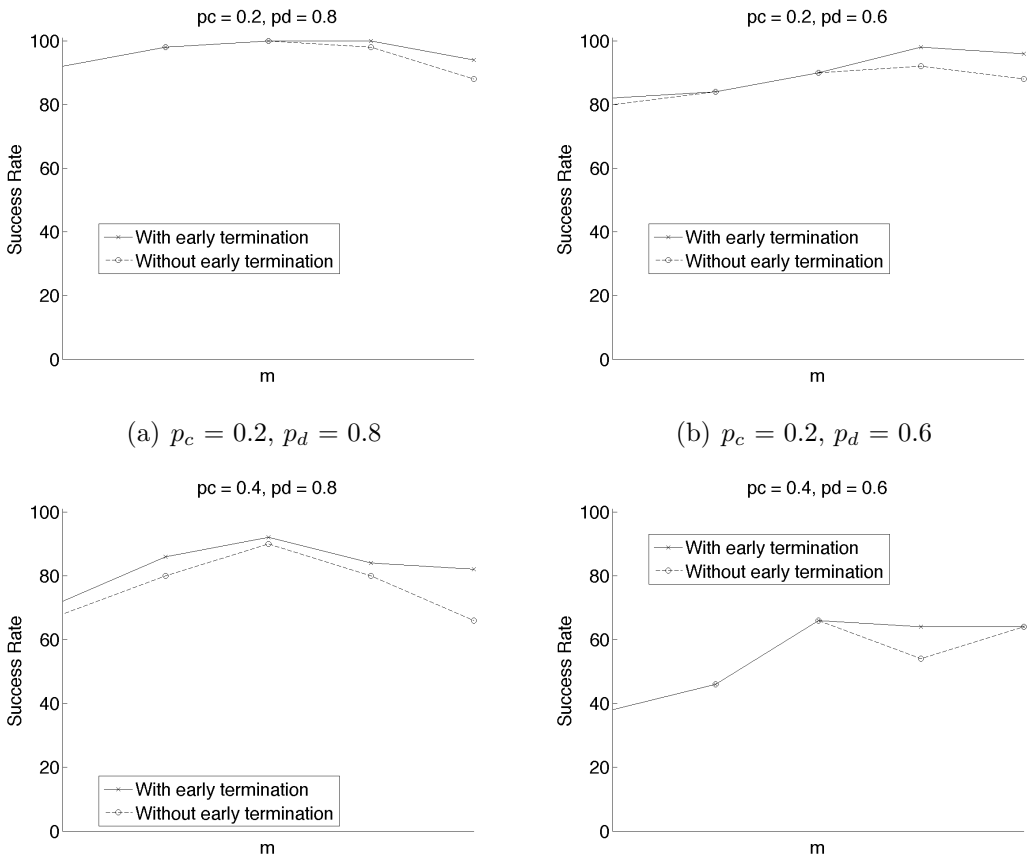


(c) $p_c = 0.4$, $p_d = 0.8$



(d) $p_c = 0.4$, $p_d = 0.6$

Figure 5.7: Comparison of the number of random starts between with and without early termination for cases that failed



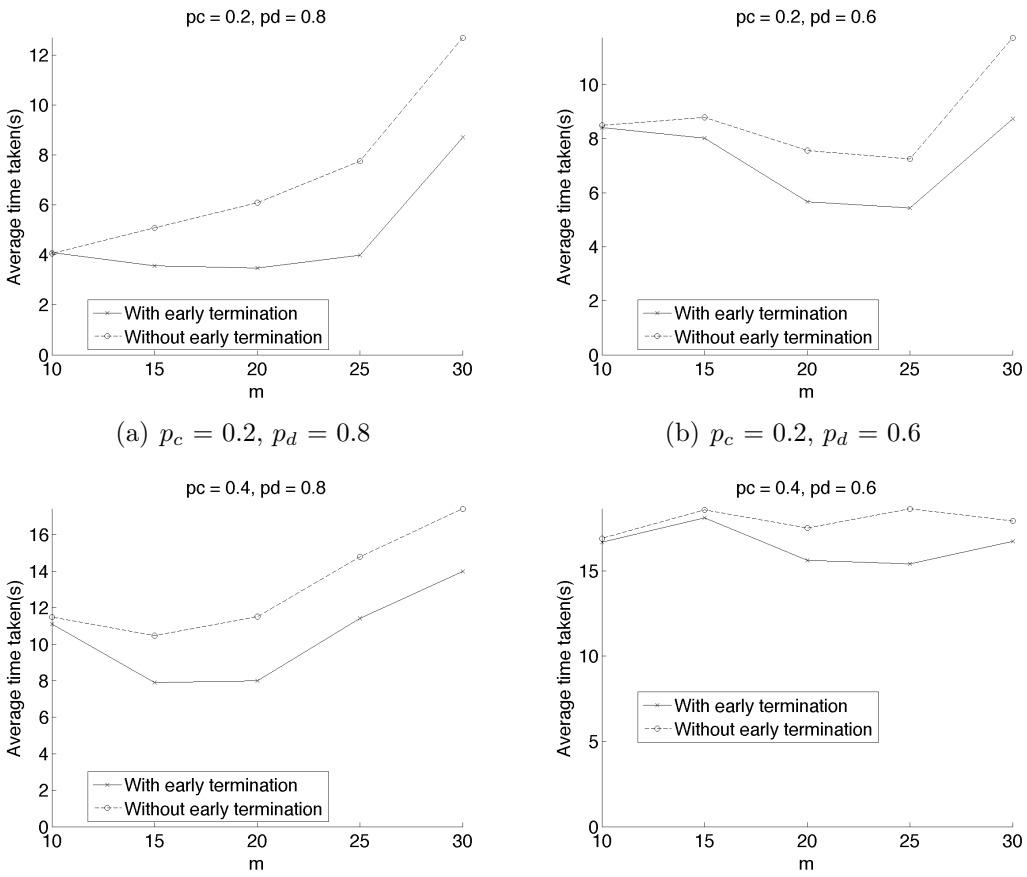
(a) $p_c = 0.2$, $p_d = 0.8$

(b) $p_c = 0.2$, $p_d = 0.6$

(c) $p_c = 0.4$, $p_d = 0.8$

(d) $p_c = 0.4$, $p_d = 0.6$

Figure 5.8: Comparison of success rate between with and without early termination



(a) $p_c = 0.2$, $p_d = 0.8$

(b) $p_c = 0.2$, $p_d = 0.6$

(c) $p_c = 0.4$, $p_d = 0.8$

(d) $p_c = 0.4$, $p_d = 0.6$

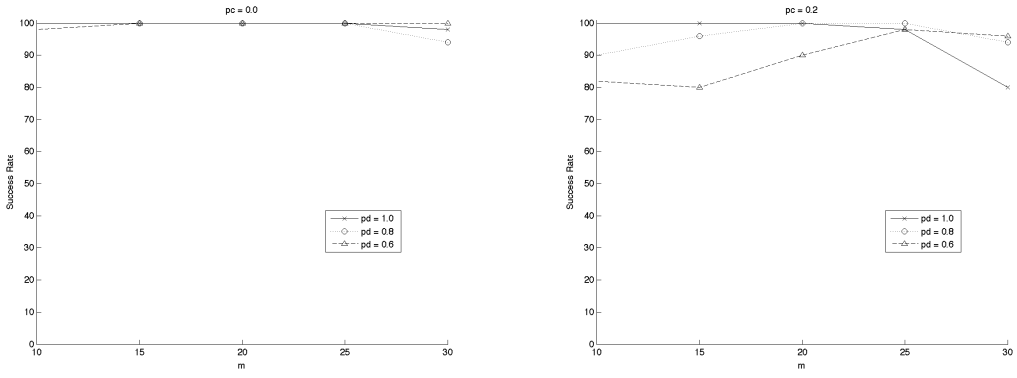
Figure 5.9: Comparison of time taken between with and without early termination

between the running time can be found in Figure 5.9.

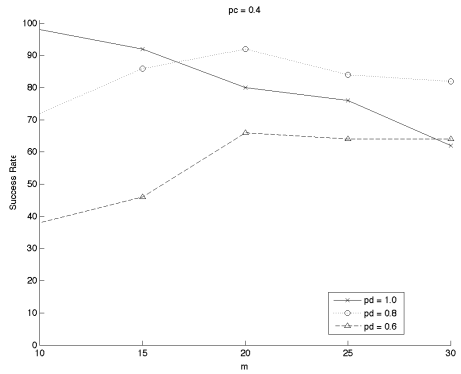
5.4 Analysis of the SoftPOSIT Method

To analyse the effectiveness of the SoftPOSIT method, the Monte Carlo test is performed under various value for m, p_d and p_c. Figure 5.10 shows the success rate of the algorithm.

It should be noted that the algorithm performs well when clutter is low. When there is no clutter, the SoftPOSIT method is able to find the correct pose in nearly all cases. When p_c is equal to 0.2, the algorithm has still has over 80% success rate even when 40% of the features are missing. This is significant because occlusion is inevitable. Under most scenarios, only a portion of the model is visible from a single point of view. Therefore, even if the feature detection method is perfect, there will still be missing features. On the other hand, it is possible to control the amount of clutter. The pose estimation may be performed in a controlled environment where the background is very simple. This may also be achieved through an image



(a) $p_c = 0.0$ (b) $p_c = 0.2$



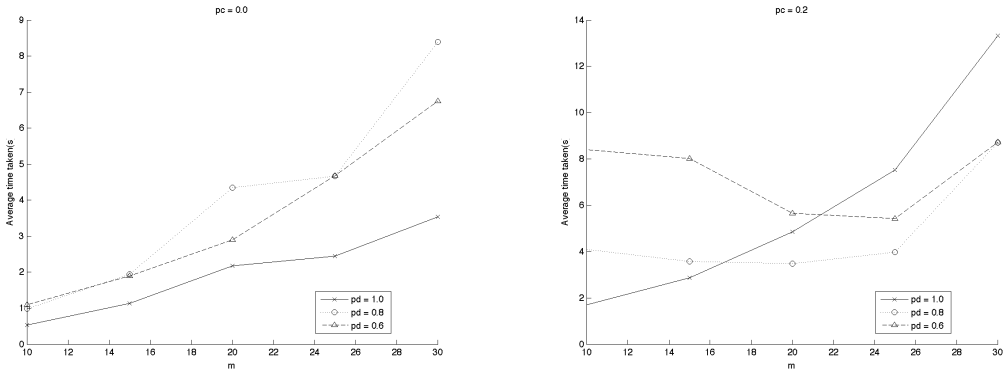
(c) $p_c = 0.4$

Figure 5.10: Success rate for SoftPOSIT method

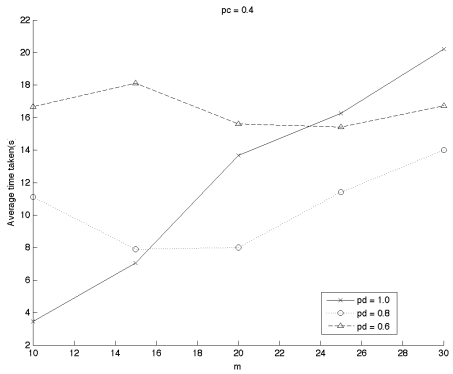
segmentation algorithm to automatically detect the area of the image that contains the model. If complete automation is not necessary, the user may be asked to manually select this area to assist pose estimation.

When clutter is high, the success rate tends to decrease as shown in Figure 5.10(c). The high clutter affects the algorithm most when there is less useful features in the image. When $p_d = 1.0$, the success rate decrease as the number of features increases. This is because increasing the number of features increases the processing time of the algorithm. Fewer random starts can be made, thus decreasing the chances of success. However, if there is enough features in the model, having some missing features actually improve the algorithm as it improves the running time as can be seen in 5.10(c). Further decreasing the number of visible features will again lead to the decrease of success rate as it increases the chance of the algorithm finding an incorrect correspondence.

Figure 5.11 shows the running time of the algorithm. The running time is correlated to the success rate. The SoftPOSIT method is allowed only 30 seconds to find the correct pose. If a test failed, it implies the full 30 seconds are used. If the pose can be found easily such as in the ideal condition where there is no



(a) $p_c = 0.0$ (b) $p_c = 0.2$



(c) $p_c = 0.4$

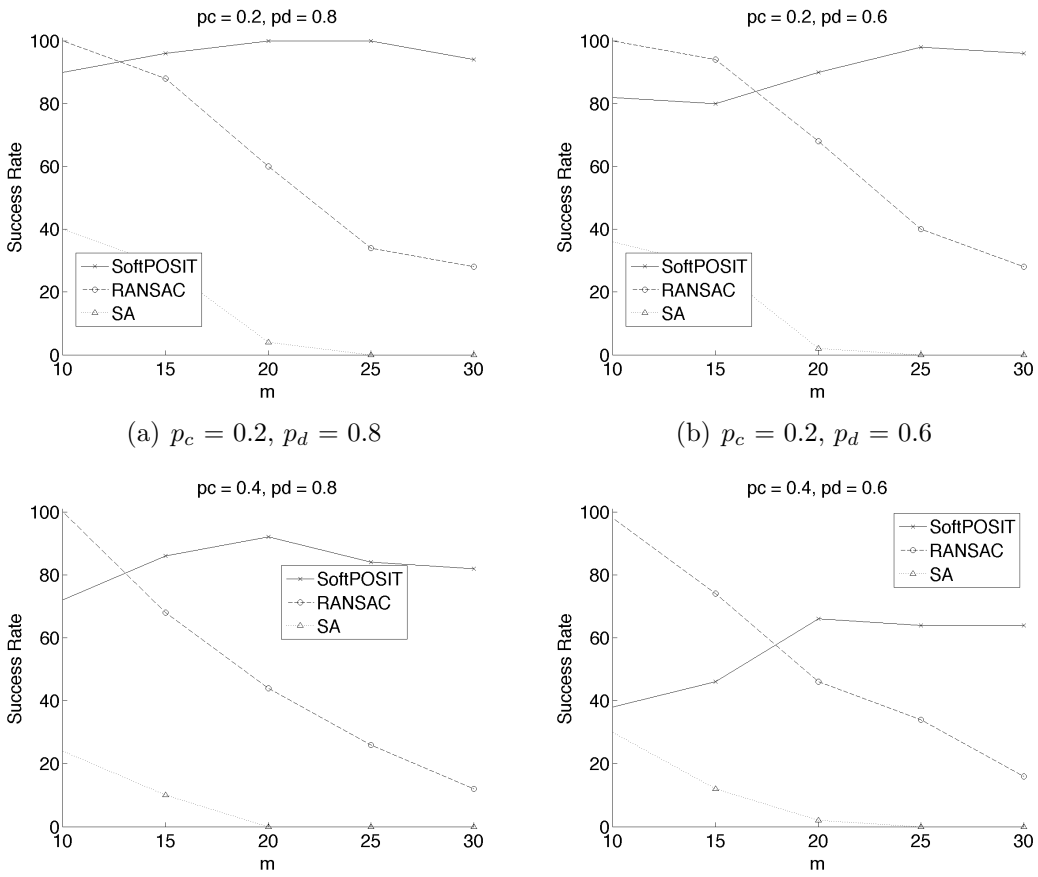
Figure 5.11: Time taken for SoftPOSIT method

occlusion or clutter, the running time increases as the number of features increases.

5.5 Comparison of Different Pose Estimation Method

A number of test are ran on the algorithms described in the thesis to determine their effectiveness. The tests are performed using the Monte Carlo method described with varying level of m, p_c and p_d. A random 3D model with a set number of lines are generated. Random translations and rotations applied to the model and projected onto an image plane. Figure 5.12 displays the success rate of each method.

In comparison with other methods, the SoftPOSIT performs well when the number of feature increases. The RANSAC favours a low feature count. It is also advantageous when the number of missing features are high. Having a high occlusion rate decreases the number of features in the image and increases the number of correspondence tested by the RANSAC method. It shows that RANSAC performs best with high occlusion and low clutter, in contrast to SoftPOSIT which tends to perform better at low occlusion and low clutter.



(a) $p_c = 0.2$, $p_d = 0.8$

(b) $p_c = 0.2$, $p_d = 0.6$

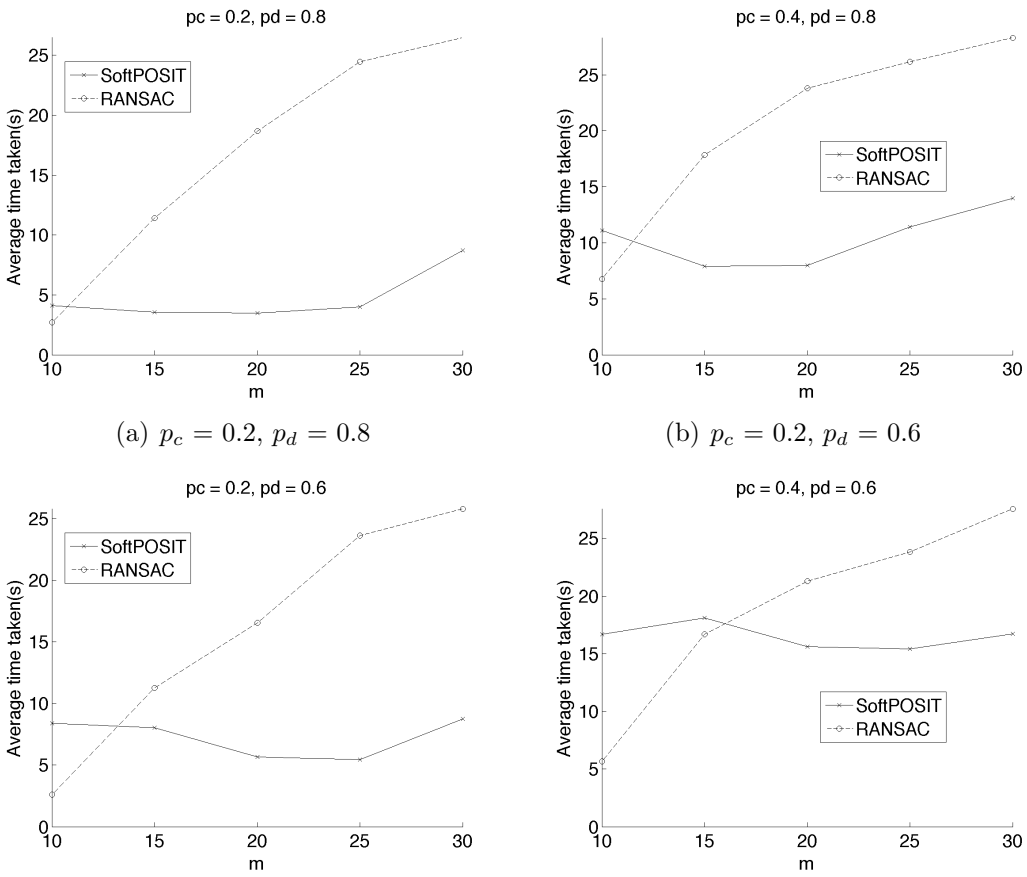
(c) $p_c = 0.4$, $p_d = 0.8$

(d) $p_c = 0.4$, $p_d = 0.6$

Figure 5.12: Comparison of success rate between different methods

The performance of RANSAC degrades rapidly as the number of feature increases. At 15 features, RANSAC still outperforms SoftPOSIT when the occlusion rate is high. At 20 features and above, SoftPOSIT performs better than RANSAC even under high occlusion condition. Figure 5.13 shows the running time of the different methods and it shows similar trend as the success rate. It can be seen that the running time of RANSAC directly correlates to the number of features and increases rapidly as the number of features increases. Whereas the relation between running time and the number of features is more obscure for the SoftPOSIT method, with the running time remaining relatively stable for the range of tests performed.

Unfortunately, the performance of simulated annealing is very unsatisfactory. The main reason for this is most likely due to the large search space for the solution in SA. For RANSAC, it is sufficient to find 3 correct correspondences. SA on the other hand attempts to find the complete correspondence as the solution. Even if it is assumed that all features have to be matched, the solution space have a size of $n!/(n-m)!$, where n is the number of image feature and m is the number of model features. The actual search space is even larger as it is possible for a model feature to be unmatched. The solution space is too large to be searched within 30 seconds,



(a) $p_c = 0.2$, $p_d = 0.8$

(b) $p_c = 0.2$, $p_d = 0.6$

(c) $p_c = 0.4$, $p_d = 0.8$

(d) $p_c = 0.4$, $p_d = 0.6$

Figure 5.13: Comparison of time taken between different methods

leading to the poor performance of SA. The running time of SA is not compared as the data is relatively meaningless. Due to the high failure rate, the average running time of the algorithm is close to the time limit of 30 seconds.

The results suggest that the SoftPOSIT algorithm is a solid algorithm under a large variety of situations. It shows its strength in the high feature count scenario. As mentioned previously, the main concern is the low clutter situations as there are various techniques to remove clutter. The SoftPOSIT algorithm outperforms other competitors in terms of both speed and success rate when the feature count is 20 or above. Considering that even a simple 3D objects like a cube contains 12 lines, it is reasonable to assume most real life object would be described by close to 20 lines or more. This means the SoftPOSIT method would be more likely to be the algorithm of choice when compared with the other two algorithms tested. Of course, it is necessary to consider the situation of which pose estimation is required. If the target model is very simple and can be described by less than 20 lines, then RANSAC may be a more suitable contender.

5.6 CAD Model Testing

In order to produce a test that closer resembles real life scenarios, a different approach is used. A number of simple CAD models are created. The model are rendered into an image to simulate a photo from camera. An example of an image is the L-shaped object shown in Figure 5.14.

A Canny edge detector[3] is used on the image to produce an edge map shown in Figure 5.15(a), and the lines are detected using the standard Hough transform [11]. The black lines in Figure 5.15(b) shows the lines the are detected from the image.

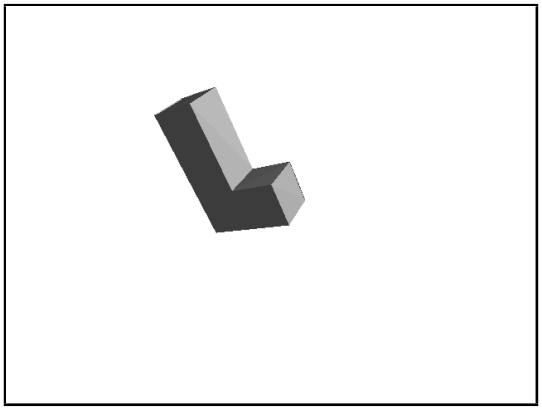
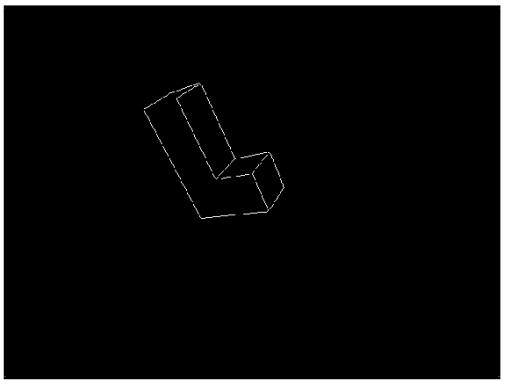
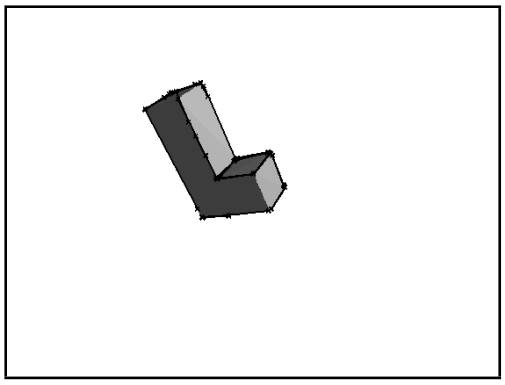


Figure 5.14: A sample of model used



(a) Edge map of the model



(b) Lines detected from the model

Figure 5.15: Edge map created by Canny edge detector and lines detected with Hough transform

It can be seen that the Canny edge detector is able to produce an accurate edge map for the image. Unfortunately, the performance of the Hough transform is less than ideal. A few obvious lines of the model are missing and more importantly, some long lines are separated into a number of small line.

These conditions makes it more difficult for the SoftPOSIT method to find the pose. An additional difficulty is present as the model look similar in appearance under different pose. For example, as the model is rotated 90° along the center line of the long side of the L shape, most of the lines on the long side will overlap. This causes the algorithm to think there is a very good correspondence for a number of lines, when there are actually incorrectly matched. The existence of many parallel lines makes it a lot easier for a model line to match to an image line incorrectly.

However, a scene like that is still simple enough, and the SoftPOSIT method is able to reliably retrieve the pose of the object within the 30 seconds limit. Figure 5.16 shows a sample of a successful pose estimate. The solid black line shows the projection of the model using the retrieved pose.

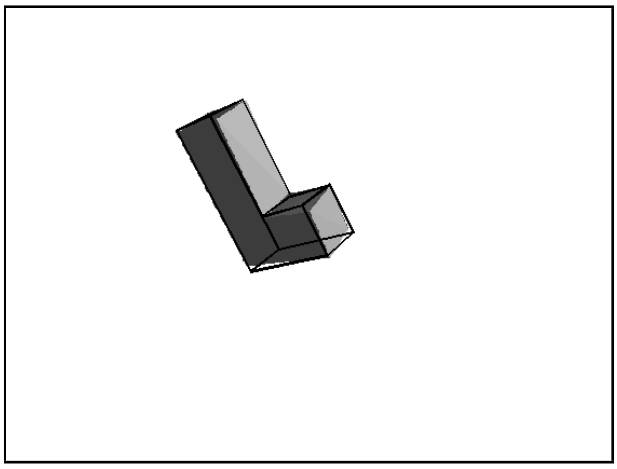


Figure 5.16: A successful pose estimation

Notice that there is some inaccuracy in the pose estimate, but it should be good enough as the initialization for visual servoing.

5.7 Real Life Image

In order to test the feasibility of the algorithm in a real life environment, a picture of a target object is taken. The photograph is shown in Figure 5.17. Lines are extracted using the Canny edge detector and Hough transform as in the previous section. Then SoftPOSIT method was executed on the detected lines in attempt to retrieve the pose.

In addition the problems encountered with the CAD model, the quality of the photograph adds an additional difficulty to the problem. This leads to a very poor line detection as shown in Figure 5.18. The white lines are the lines that were detected from the image. Some of the edges are missing and there are cases where multiple lines are detected when it should have been a single line. Clearly, more research needs to be spent on finding an optimal line detection algorithm.

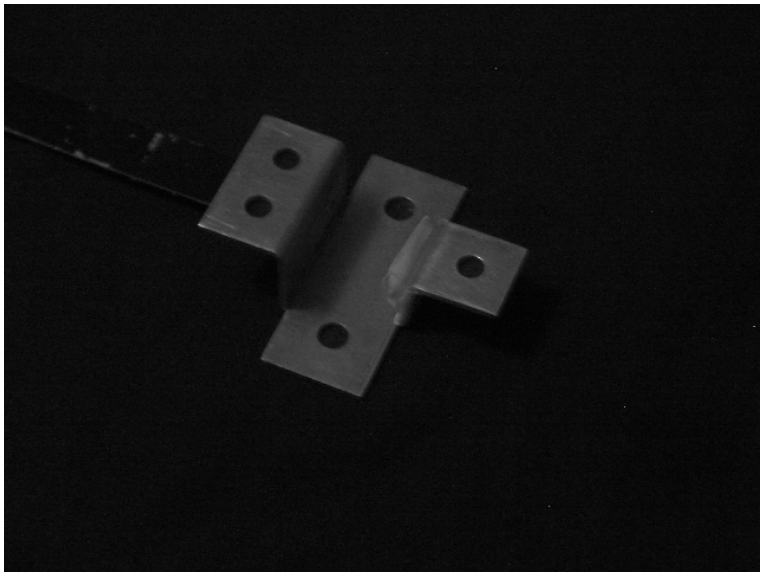


Figure 5.17: Photograph of a model

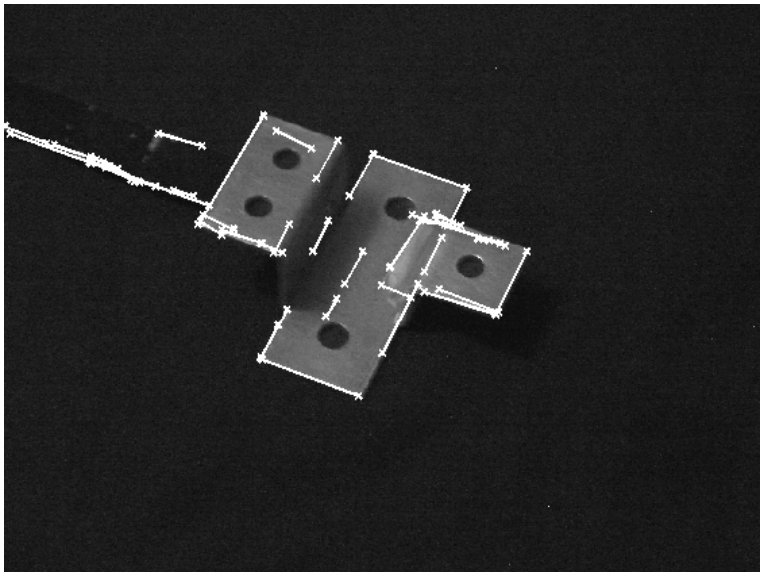


Figure 5.18: Line detected from the photograph

To prove that the SoftPOSIT method works if such algorithm is found, the line detection is performed manually by a human being. Figure 5.19 shows the manually detected lines. Figure 5.20 shows the correct pose found by the algorithm.

This test shows that the SoftPOSIT method is limited by the performance of the line detection method. It is able to retrieve a good pose if the lines are detected correctly. A poor line detector often results in missing lines or addition clutter, both of which impact the performance of the SoftPOSIT algorithm. More reliable algorithms should exist, but more research is needed to find one that is suitable for solving the pose estimation problem.

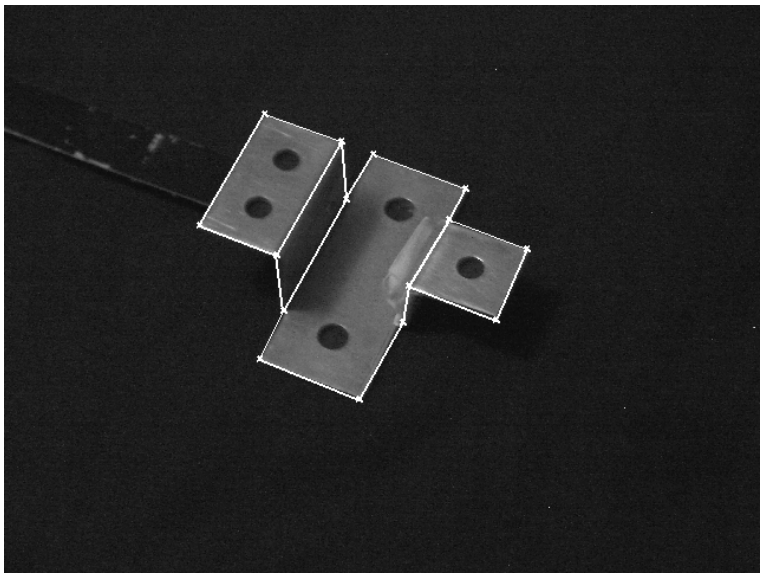


Figure 5.19: Ideal line detection

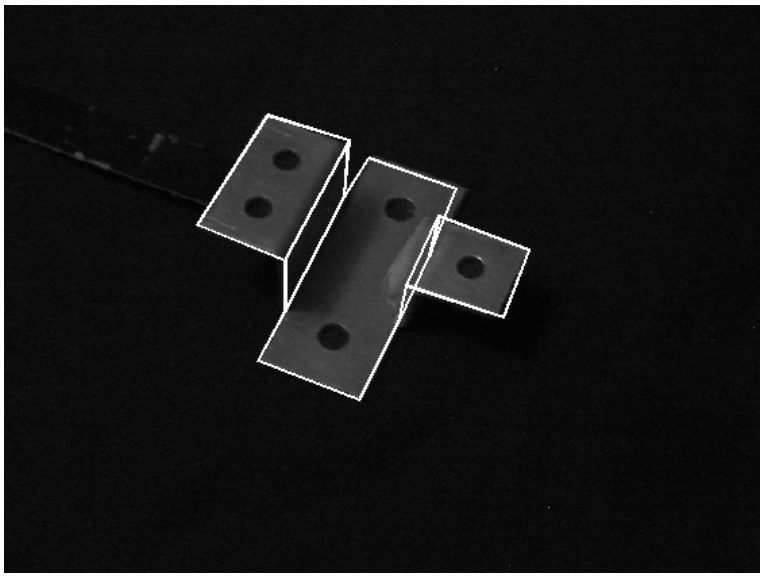


Figure 5.20: Correct pose of the model

Chapter 6

Conclusions and Further Work

6.1 Conclusions

In this research, various methods for performing pose estimation were examined. Specifically, the purpose of the pose estimation is designed for initialization of a visual servoing system. The simultaneous pose and correspondence problem is separated into the pose estimation problem and the correspondence problem. The pose estimation with correspondence problem was first investigated.

The Gauss-Newton method and the POSIT method were examined as the solution to the pose estimation problem. Iterative methods were chosen as they can be modified to solve for the pose when the correspondence is unknown. Experiments have shown that POSIT method outperforms the Gauss-Newton method in terms of both the speed and accuracy. It is capable of retrieving 100% of the pose. This promising results lead to further research with the POSIT method as the base of the solution.

The SoftPOSIT method became the focus of the research. It is a method that alternates between solving for the pose problem and then solving the correspondence. An initial pose is assumed. Using the initial pose, a correspondence matrix is calculated based on the distances between an image feature and the projected feature of the model. The Sinkhorn's method is used to ensure the matrix is doubly stochastic. With this correspondence, a pose is calculated using the POSIT method, thus giving better pose estimation. The new pose gives rise to a new correspondence and the process is repeated until the algorithm converges.

To improve upon this method, a new early termination condition is developed. This forces another pose to be examined if it seems unlikely that the algorithm will converge to a correct pose. The termination condition checks the number of model features that are within certain distance from an image feature. If this number is below a certain value, the algorithm terminates and starts with another initial guess. As the algorithm runs, the distance between features should decrease, thus the threshold distance will also decrease. The use of this termination condition

allows a slight increase in the success rate of the algorithm and the significantly improves upon the running time.

An investigation was carried out to determine the type of feature to be used. The features that were considered were points and lines. A Monte Carlo test was performed. A random model made of either line or point features was generated. The model was transformed and projected onto a 2D plane. Some features were removed while some other extra erroneous features were added. The SoftPOSIT method was executed in attempt to find the transformation that was performed. The test results show that using line features is preferable as it reduces the number of iterations required for the algorithm to achieve the same accuracy. It also reduces the chance of false positives.

The SoftPOSIT method is also tested against alternative methods which includes the RANSAC algorithm and simulated annealing. The Monte Carlo method is used again to for the purpose of testing. The SoftPOSIT method is proven to be more successful in finding the correct pose under high feature count environment, whereas RANSAC performs slightly better when the feature count is low. Simulated annealing underperformed in all scenarios tested.

While both the SoftPOSIT method and the RANSAC method have solid performance, application to real world environment is limited by the performance of the feature detection algorithm. More research is required to find a reliable method to detect the required feature.

6.2 Further Work

While results suggest the SoftPOSIT algorithm is a decent method of retrieving pose in a relatively short time, there are many areas which can be improved on. An obvious one is the physical implementation of the work in a visual servoing system. The tests performed in the research was implemented in MATLAB as a simulation. Implementing and optimizing the algorithm in a native machine language should theoretically improve performance.

Another improvement to consider is the calculation of occlusion. The calculation of the correspondence matrix is based upon the projection of the model on a 2D plane. Typically, some features will be occluded by the object itself. Despite this, the correspondence calculation does not discount the occluded features. This increases the chance of a feature mismatch and reduces the convergence rate. Given certain pose, it is not difficult to determine which features should have been occluded. Once the occluded features are determined, they can be eliminated from the correspondence calculation. Investigation should be carried out to determine how this will affect the effectiveness of the algorithm. However, this will most likely require the use of a graphics processing unit or specialized hardware in order to achieve a satisfactory performance.

The feature choice is also a point for consideration. A typical real life object contains many curves rather than straight lines and points. Thus, this algorithm will not be suitable for many objects. If features such as curves and texture can be incorporated, it may allow pose estimation to be performed on a much greater variety of items. It should also be possible to incorporate different types of features into the algorithm instead of using just a single type. Multiple features types allows a target to be described with higher accuracy.

Another way to consider multiple feature types is how they interact with each other. Two corners may be connected by an edge. Therefore, knowing the correspondence of one feature may lead to the correspondence of another feature. It may be possible to determine a much better correspondence by studying how the features connect to each other in the image.

For this thesis, a single image of the model is used. It is possible for the system to take a series of images and produce a 3D model from multiple images. Having a 3D representation of the scene make the pose estimation process easier as the perspective projection of the model do not need to be considered. However, additional time is require to take multiple images and the 3D representation may not be accurate. It would be worth comparing the difference in the performance of the two different approaches.

When a feature is detected, much information about the feature is dropped. For example, the color or texture around a feature may allow the correspondence to be identified. The angles between the lines that forms a corner can provide indication about the orientation of the model. Ignoring such details removes some information that could contribute to retrieving the pose of the object.

Finally, as mentioned through the thesis, the reliability of the feature detection algorithm is an important factor for pose estimation. Such algorithm may already exist, and further investigation should be performed in order to produce a capable visual servoing system.

Appendix A - Minimization for Weighted Least Squares

Suppose, given a linear system such that there is a set of data found upon n observations, $\mathbf{y}$, and a set of m parameters, β, the error, $\mathbf{r}$, is defined by

$$\mathbf{r} = \mathbf{X}\beta - \mathbf{y} \tag{1}$$

where $\mathbf{X}$ is a $n \times m$ matrix which defines the model of the linear system. If the goal is to find β such that

$$\sum_{i=1}^{n} r_i^2 \tag{2}$$

is the minimized, it is well known that β would satisfy the equation

$$\mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^T\mathbf{y} \tag{3}$$

If some of the error is known to have higher certainty, a weighted sum of square can be used. The goal would be to minimize

$$\sum_{i=1}^{n} w_i r_i^2 \tag{4}$$

where w_i represents the weighting of error i. Then, β will satisfy the equation

$$(\mathbf{X}^T\mathbf{W}\mathbf{X})\beta = \mathbf{X}^T\mathbf{W}\mathbf{y} \tag{5}$$

where $\mathbf{W}$ is a diagonal matrix with $W_{ii} = w_i$. This can be rewritten as

$$\left(\sum_{i=1} w_i \begin{bmatrix} X_{i1}X_{i1} & X_{i1}X_{i2} & \cdots & X_{i1}X_{im} \\ X_{i2}X_{i1} & X_{i2}X_{i2} & \cdots & X_{i2}X_{im} \\ \vdots & \vdots & \ddots & \vdots \\ X_{in}X_{i1} & X_{in}X_{i2} & \cdots & X_{in}X_{im} \end{bmatrix}\right)\beta = \sum_{i=1}^{m}\sum_{j=1}^{n} \begin{bmatrix} w_i y_j X_{j1} \\ w_i y_j X_{j2} \\ \vdots \\ w_i y_j X_{jm} \end{bmatrix} \tag{6}$$

Let $\mathbf{X} = \begin{bmatrix} \mathbf{X_1} & \mathbf{X_2} & \cdots & \mathbf{X_m} \end{bmatrix}^T$ gives

$$\left(\sum_{i=1}^{n} w_i \mathbf{X_i} \mathbf{X_i}^T\right)\beta = \sum_{i=1}^{n} w_i y_i \mathbf{X_i} \tag{7}$$

$$\beta = \left(\sum_{i=1}^{n} w_i \mathbf{X_i} \mathbf{X_i}^T\right)^{-1} \sum_{i=1}^{n} w_i y_i \mathbf{X_i} \tag{8}$$

Appendix B - Solution to P3P Problem

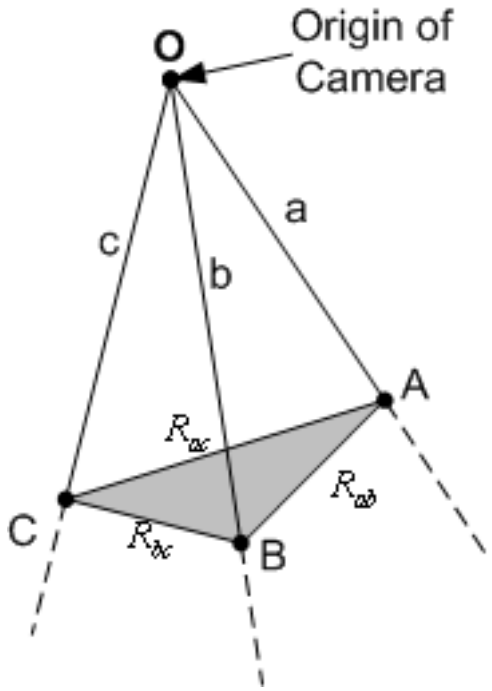


Figure 1: Geometry of the P3P problem

The solution to the P3P problem is presented for the convenience of the reader. It can also found in Fishcler[12]. Given the three sides of the base of a tetrahedron (R_{ab}, R_{bc}, R_{ac}) and the corresponding angles to each side (θ_{ab}, θ_{bc}, θ_{ac}) as shown in Figure 1, the problem is to find the lengths of the three remaining sides of the tetrahedron (a, b, c).

A solution can be obtained by solving the following system of equations:

$$(R_{ab})^2 = a^2 + b^2 - 2ab\cos(\theta_{ab}) \quad (9)$$

$$(R_{ac})^2 = a^2 + c^2 - 2ac\cos(\theta_{ac}) \quad (10)$$

$$(R_{bc})^2 = b^2 + c^2 - 2bc\cos(\theta_{bc}) \quad (11)$$

Let $x = b/a$ and $y = c/a$ gives:

$$\begin{aligned}
(R_{ab})^2 &= a^2 + x^2a^2 - 2a^2x\cos(\theta_{ab}) && (12)\\
(R_{ac})^2 &= a^2 + y^2a^2 - 2a^2y\cos(\theta_{ac}) && (13)\\
(R_{bc})^2 &= x^2a^2 + y^2a^2 - 2a^2xy\cos(\theta_{bc}) && (14)
\end{aligned}$$

Equations 12 and 14 give:

$$(R_{bc})^2[1 + x^2 - 2x\cos(\theta_{ab})] = (R_{ab})^2[x^2 + y^2 - 2xy\cos(\theta_{bc})] \quad (15)$$

Equations 13 and 14 give:

$$(R_{bc})^2[1 + y^2 - 2y\cos(\theta_{ac})] = (R_{ac})^2[x^2 + y^2 - 2xy\cos(\theta_{bc})] \quad (16)$$

Let $K_1 = (R_{bc})^2/(R_{ac})^2$ and $K_2 = (R_{bc})^2/(R_{ab})^2$. Substituting into equations 16 and 15 gives:

$$\begin{aligned}
0 &= y^2(1 - K_1) + 2[K_1\cos(\theta_{ac})]y - x\cos(\theta_{bc}) + x^2 - K_1 && (17)\\
0 &= y^2 + 2[-x\cos(\theta_{bc})]y + [x^2(1 - K_2) + 2xK_2\cos(\theta_{ab})] && (18)
\end{aligned}$$

Now, 17 and 18 have the form:

$$\begin{aligned}
0 &= my^2 + py + q, && (19)\\
0 &= m'y^2 + p'y + q', && (20)\\
& && (21)
\end{aligned}$$

where

$$\begin{aligned}
m &= (1 - K_1)\\
p &= 2K_1\cos(\theta_{ac})\\
q &= -x\cos(\theta_{bc}) + x^2 - K_1\\
m' &= 1\\
p' &= -2x\cos(\theta_{bc})\\
q' &= x^2(1 - K_2) + 2xK_2\cos(\theta_{ab})
\end{aligned}$$

Multiply 19 by m' and 20 by m and subtracting gives:

$$0 = (pm' - p'm)(y) + (m'q - mq') \quad (22)$$

Multiply 19 by q' and 20 by q, subtracting, then divide by y gives:

$$0 = (m'q - mq')(y) + (p'q - pq') \quad (23)$$

Multiply Equation 22 by $(m'q - mq')$ and Equation 23 by $(pm' - p'm)$, and subtract to obtain:

$$0 = (m'q - mq')^2 - (pm' - p'm)(p'q - pq') \quad (24)$$

Expanding the equation and grouping terms gives a quartic polynomial in x:

$$0 = G_4x^4 + G_3x^3 + G_2x^2 + G_1x^1 + G_0 \tag{25}$$

where

$$\begin{aligned}
G_4 &= (K_1K_2 - K_1 - K_2)^2 - 4K_1K_2\cos(\theta_{bc}) \\
G_3 &= 4(K_1K_2 - K_1 - K_2)(K_2)(1 - K_1)\cos(\theta_{ab}) \\
&\quad +4K_1\cos(\theta_{bc})[(K_1K_2 + K_2 - K_1)\cos(\theta_{ac}) \\
&\quad +2K_2\cos(\theta_{ab})\cos(\theta_{bc})] \\
G_2 &= [2K_2(1 - K_1)\cos(\theta_{ab})]^2 + 2(K_1K_2 + K_1 - K_2)(K_1K_2 - K_1 - K_2) \\
&\quad +4K_1[(K_1 - K_2)\cos^2(\theta_{bc}) + (1 - K_2)(K_1)\cos^2(\theta_{ac}) \\
&\quad -2K_2(1 + K_1)\cos(\theta_{ab})\cos(\theta_{ac})cos(\theta_{bc})] \\
G_1 &= 4(K_1K_2 + K_1 - K_2)(K_2)(1 - K_1)\cos(\theta_{ab}) \\
&\quad +4K_1[(K_1K_2 - K_1 + K_2)\cos(\theta_{ac})\cos(\theta_{bc}) \\
&\quad +2K_1K_2\cos(\theta_{ab})\cos^2(\theta_{ac})] \\
G_O &= (K_1K_2 + K_1 - K_2)^2 - 4(K_1^2)(K_2)\cos(\theta_{ac})^2
\end{aligned} \tag{26}$$

Equation 25 is a quartic equation can be solved with closed form. For each positive real root, a single positive real value of each of sides a and b can be determined. From Equation 12:

$$a = \frac{R_a b}{\sqrt{x^2 - 2x\cos(\theta_{ab}) + 1}} \tag{27}$$

Recall that $x = a/b$, thus

$$b = ax \tag{28}$$

If $m'q \neq mq'$, then Equation 23 gives

$$y = \frac{p'q - pq'}{mq' - m'q} \tag{29}$$

If $m'q = mq'$, then Equation 29 is undefined and Equation 13 gives two values of y:

$$y = \cos(\theta_{ac}) \pm \sqrt{\cos^2(\theta_{ac}) + \frac{(R_{ac})^2 - a^2}{a^2}} \tag{30}$$

Each value of y gives a value of c:

$$c = ya \tag{31}$$

When values of y are obtained from Equation 13 rather than Equation 29, the resulting solutions must be shown to satisfy Equation 11 before it can be accepted.

References

[1] H.G. Barrow, R.C. Bolles J.M. Tenenbaum, and H.C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. *International Joint Conference on Artificial Intelligence*, 2:659–663, 1977. 8

[2] Paul Besl. A method for registration of 3-D shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–255, February 1992. 9

[3] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–697, November 1986. 6, 48

[4] F. Chaumette and S. Hutchinson. Visual servo control, part i: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, December 2006. 2

[5] F. Chaumette and S. Hutchinson. Visual servo control, part ii: Advanced approaches. *IEEE Robotics and Automation Magazine*, 14(1):109–118, March 2007. 2

[6] Haili Chiu and Anand Rangaranjan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, October 2003. 9

[7] Philip David, Daniel DeMenthon, Ramani Duraiswami, and Hanan Samet. Simultaneous pose and correspondence determination using line features. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:424–431, June 2003. 26

[8] Philip David, Daniel Dementhon, Ramani Duraiswami, and Hanan Samet. SoftPOSIT: Simultaneous pose and correspondence determination. *International Journal of Computer Vision*, 59:259–284, September 2004. 9, 22, 28, 34

[9] Daniel Dementhon and Larry Davis. New exact and approximate solutions of the three-point perspective problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(17):1100–1105, November 1992. 8

[10] Daniel Dementhon and Larry Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, May 1995. 8, 9, 11, 16

[11] Richard Duda and Peter Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1), January 1972. 6, 48

[12] Martin A. Fishcler and Robert C. Bolles. Random sampling consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communication of the ACM*, 24(6):381–395, June 1981. 8, 9, 30, 31, 56

[13] Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu, and Eric Mjolsness. New algorithm for 2D and 3D point matching: Pose estimation and correspondence. *Pattern Recognition*, 31:956–964, 1998. 9, 22

[14] X. C. He and N. H. C. Yung. Curvature scale space corner detector with adaptive threshold and dynamic region of support. *International Conference on Pattern Recognition*, 2:791–794, August 2004. 6

[15] Jiawei Hong and Xiaonan Tan. A new approach to point pattern matching. *International Conference on Pattern Recognition*, 1:82–84, 1988. 8

[16] Berthold K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4:629–642, April 1987. 8

[17] Berthold K.P. Horn, Hugh M. Hildon, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5:1127–1135, July 1988. 8

[18] C.S. Langley and G.M.T. D'Eleuterio. Pose estimation for fixtureless assembly using a feature cmac neural network. *Assembly Automation*, 25(3):204–216, May 2000. 1

[19] David Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:441–450, May 1991. 8, 11, 15

[20] David Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 2(9):1150, September 1999. 9

[21] Feng Lu and Evangolos Milios. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, March 1997. 1

[22] Donald Marquardt. An algorithm for least-square estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963. 16

[23] Donald Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963. 15

[24] Nicholas Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, September 1949. 20

[25] A. Prusak, O. Melynchuk, H. Roth, I. Schiller, and R. Koch. Pose estimation and map building with a time-of-flight-camera for robot navigation. *International Journal of Intelligent Systems Technologies and Applications*, 5(3):355–364, 2008. 1

[26] Szymon Rusinkiewica. Efficient vairants of the icp algorithm. *International Conference on 3D Digital Imaging and Modeling*, pages 145–152, May 2001. 9

[27] Guy Scott and H. Christopher Longuet-Higgins. An algorithm for associating the features of two images. *Proceedings of Royal Society London B*, 244:21–26, February 1991. 9

[28] Larry Shapiro and Michael Brady. Feature-based correspondence: An eigenvector approach. *Image and Vision Computing*, 10(5):283–288, June 1992. 9

[29] Linda Shapiro and George Stockman. *Computer Vision*. Prentice Hall, Feburary 2001. 4

[30] Richard Sinkhorn. A relation between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, Jun 1964. 24

[31] J.P. Pascual Starink and Eric Backer. Finding point correspondence using simulated annealing. *Pattern Recognition*, 28(2):231–240, July 1994. 31

[32] Shinji Umeyama. Least-square estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991. 8

[33] William Wilson, Carol Hulls, and Graham Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transaction on Robotics and Automation*, 12(5):684–696, October 1996. 2, 8